
CALL MSX
Número 7 · Diciembre 2006
Cómo pasar
Ghost House
Entrevista
Tsujikawa
Artículo
Aritmética
de punto fijo
[redacted]
¡Una secuela esperada!
MANBOW 2

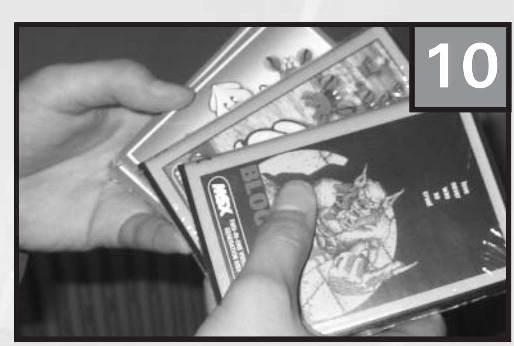
NÚMERO 7

DICIEMBRE / 2006



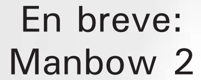
Sumario

secciones

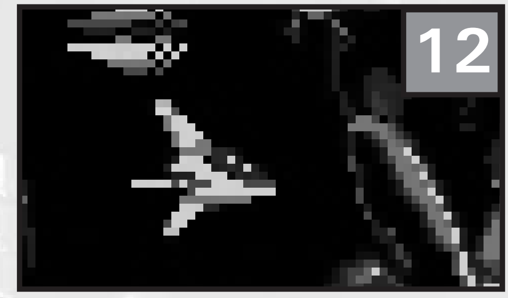


Eventos: 29a Ru de Barcelona

Una reunión más en la capital catalana que no despertó el mismo interés que su predecesora por la escasez de novedades. Aun así, hubo motivos de sobra para acercarse.



En breve: Manbow 2

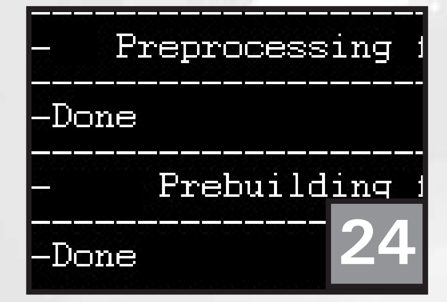


Manbow 2 será una realidad muy pronto. Aquí encontrarás una introducción a lo que promete ser un juegazo.



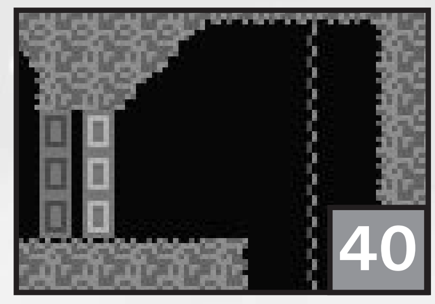
Ya disponible: Operation Wolf

Es cierto, nunca tuvimos una versión de la recreativa como merecíamos. El grupo Toybox nos ofrece una nueva versión más acorde a las posibilidades de nuestra máquina.



Artículo: Desarrollo cruzado en PC para MSX

Programar para MSX utilizando el PC.



Software Amateur: Spelunkers

En esta sección veremos juegos y demos creadas por usuarios. Spelunkers es buena muestra de ello.



Cómo pasar ...: Ghost House

Si se te atascaba el mítico juego de Casio te ofrecemos una guía para poder llegarte al final.

Redacción

Francisco Álvarez
Roberto Álvarez
Óscar Centelles
David Lucena

Ilustraciones

Roberto Álvarez

Maquetación

Francisco Álvarez
Roberto Álvarez
Óscar Centelles

Colaboran

David Fernández
Avelino Herrera
Juan Luis Martínez
Juan Miguel Ortuño
Manuel Pazos
Armando Pérez

Contacto

http://callmsx.gabiot.com
por Avelino Herrera

EDITORIAL

¡A TODA MÁQUINA!

Con el nombre de aquel famoso pack de Dinamic podríamos resumir la gran actividad dentro de la escena del MSX. Y es que multitud de títulos aparecen día tras día, muchos impulsados por la conocida MSX Dev y otros desarrollados independientemente gracias a ese espíritu tan poderoso que crece y no deja de sorprendernos. Es tal el volumen de novedades que hemos tenido que crear nuevos artículos o realizar modificaciones a última hora, justo antes de enviar el magazine a la imprenta. Por un lado pensábamos: ¡Dejadnos acabar la revista en paz! pero por el otro pedíamos más y más novedades. El detalle a destacar es que últimamente no tenemos dificultades para llenar la revista con todo el material que aparece, algo que denota nuevamente la actividad actual del sistema.

En la era de Internet donde todo el mundo puede descargar cualquier juego y que el soporte físico, en muchos casos, está en un plano secundario o inexistente, el MSX vuelve a los orígenes dando soporte nuevamente a la cinta o a nuevos y avanzados cartuchos; todo ello culminado por un gran logro para la escena: el One Chip MSX, un sistema MSX completo fruto de la insistencia y perserverancia de muchos usuarios de MSX en todo el mundo que lo han apoyado.

Una vez más queremos agradecer la gran ayuda prestada por nuestros habituales colaboradores sin los cuales esta revista difícilmente se encontraría en vuestras manos. Un trabajo totalmente voluntario y que refuerza ese espíritu y esa manera que tenemos los usuarios del MSX de hacer las cosas. También animamos al resto de la gente a colaborar con la revista, cualquier idea, trabajo u opinión será bienvenida.

Nos despedimos felicitando las fiestas y que el 2007 nos traiga más alegrías MSXeras si es posible.

¡Hasta la próxima!

El equipo de Call MSX.

¡LLEGA POR FIN EL NUEVO MSX!



¿Impaciencia? ¿Alegría contenida? ¿Cómo podríamos definir el estado de ánimo en el que nos hallamos mientras escribimos estas líneas? El One Chip MSX ya está a la venta, eso es algo que creíamos impensable hace unos meses o años. Pensábamos que iban a volver a jugar con nosotros, con nuestros sentimientos, pero verlo para creerlo, definitivamente se vende. Ahora a los usuarios de fuera de Japón nos toca esperar un poco más para hacernos con él. Hasta entonces, a morderse las uñas.

Los usuarios japoneses están de enhorabuena porque el nuevo One Chip MSX ya está disponible. La compañía D4 se ha animado a fabricar este nuevo dispositivo y a lanzarlo al mercado al precio de 20790 yens más gastos de envío. Asimismo y tras realizar algunas modificaciones para cumplir con las exigencias del mercado europeo, la compañía Bazix se encarga de su distribución fuera de Japón.

Este nuevo ordenador equivale a un MSX2 con interesantes mejoras de serie entre las que podemos destacar el lector de tarjetas SD/MMC basado en el mismo sistema que utiliza el Mega-SCSI original con soporte nativo FAT16, y sonido MSXMusic y SCC+ incorporado de serie. Originalmente el One Chip MSX incorporaba 256 Kb pero finalmente se ha decidido aumentar dicha memoria hasta 1 Mb de RAM.

El aspecto externo es muy bueno, una carcasa de color azul transparente nos permite distinguir la circuitería y el chip Altera Cyclone, el auténtico corazón de la máquina, además queda muy llamativo el movimiento de los leds rojos, de izquierda a derecha. También podemos observar los dos slots perfectamente protegidos por unas pequeñas puertas basculantes y un par de botones, el de encendido y el reset. Echando un vistazo a los conectores podemos ver en la parte posterior los típicos conectores de video y audio estéreo (RCA), S-Video, VGA y alimentación. En la parte frontal podemos hallar un par de entradas USB que en un futuro podrán ser utilizadas, dos puertos serie y el fantástico lector de tarjetas SD, totalmente operativo.

Ya que el One Chip MSX está basado en el Altera Cyclone, un chip FPGA totalmente reprogramable, podemos variar su estructura para que se comporte de manera diferente, pudiendo cambiar la lógica para que equivalga a otro procesador; siempre y cuando no sobrepasemos la capacidad o el número de puertas reprogramables de dicha FPGA. Esto es muy interesante ya que se pueden realizar actualizaciones o ampliaciones del sistema simplemente modificando el código interno del chip principal.

Según informaciones hechas públicas, el código para emular cada chip de nuestro MSX está lejos de estar optimizado al 100%, lo que deja una puerta abierta a estos futuros “upgrades” de Hardware. Si el futuro desarrollador de Hardware para el MSX One Chip quiere mostrar su destreza con el código VHDL será mejor que adquiera (o se construya) el cable necesario, pues podría tener resultados imprevistos y dejar su máquina inservible por un tiempo.

CARACTERÍSTICAS TÉCNICAS :

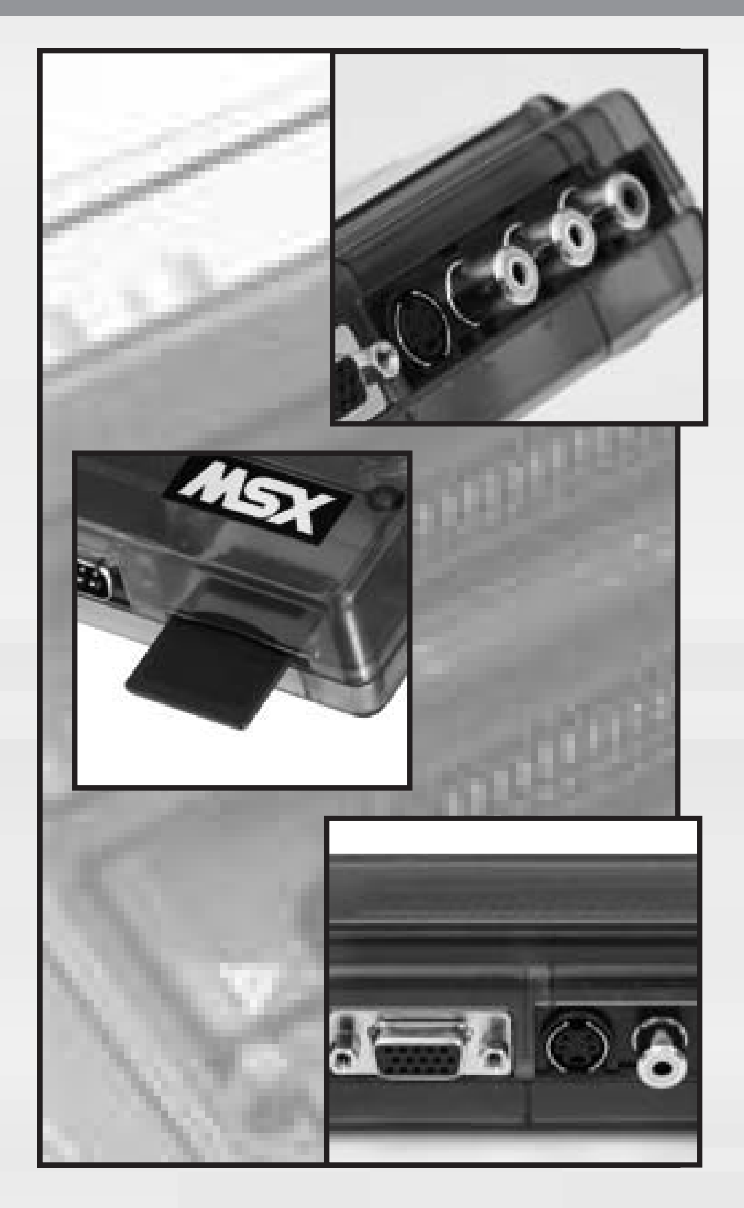
- MSX2 con 1024kB RAM
- Soporte Kanji
- Soporte MSX-DOS2
- Conexión PS/2
- 2 puertos de joystick MSX
- 2 slots de cartuchos MSX (soporte de 12 voltios)
- Slot de tarjeta SD/MMC con soporte nativo para FAT16 bajo MSX-DOS 2
- Salida de vídeo compuesto y SVHS
- Salida para monitor VGA
- 2 salidas de audio
- Pin de E/S para la FPGA (40 pins y 10 pins)
- 2 puertos USB
- 32MB SDRAM

One Chip MSX

Tuvimos tiempo hasta el pasado 21 de Noviembre para reservar nuestro One Chip MSX, ya que al parecer no habrá una segunda venta de los mismos. El "package" del One Chip MSX contendrá:

MATERIAL INCLUIDO :

- 1 One Chip MSX
- Manual de usuario en inglés
- Código fuente en VHDL de la configuración MSX2
- CDROM con fuentes y programas:
    - Altera Quartus II - Desarrollar en VHDL
    - PLDLOAD / PLDSAVE - Cargar/Grabar la FPGA
    - EP, FDLOAD, FDSAVE - Utilidades para DSKs
- Adaptador de 220V
- Esquemas de la placa del One Chip MSX
- Anteproyectos de la carcasa del One Chip MSX





Para finalizar, sólo confirmar que One Chip MSX, sumando gastos de envío y el famoso VAT nos sale a los españoles a un precio que ronda los 240 euros. Posiblemente la cantidad sea un poco más elevada que la que pagarán en el mercado japonés, pero por todo lo visto, merecerá la pena.

4º Número de Moai-Tech

Ya podéis visitar en internet la cuarta entrega del webzine de MSX en castellano.

A destacar en este número el comentario del primer cartucho del milenio de MSX: Ink Exxon Surfing y la entrevista al grupo holandés Delta Soft, creadores del Konami Quiz 2, los cuales veremos que tienen una visión del futuro del sistema que a más de uno les chocará.

Como siempre somos fans de la sección Curiosidades y esta vez nos ha parecido un poco corta, nos hubiera gustado que profundizaran un poco más en esos aparatos koreanos tan sorprendentes como el Kobo.

Después de bastante tiempo nos muestran un CD musical relacionado con el MSX, está bien dar de vez en cuando un placer al oído.

También destacar la mención al anime del cual Konami hizo un soberbio juegazo, Hinotori. Para los fans de este juego nos facilitan el enlace para descargar la película desde el programa de PC eMule.

http://www.ivanpriego.com/0_moai/

MSX TYPHOON

MSX Typhoon es un disco compacto musical del artista japonés Araki Kenta que produjo 360º Records en el 2005 y que ha llegado hace unos días a nuestra redacción. El mismo presenta un acabado profesional al igual que podríamos encontrar en cualquier producto de venta en tiendas. Esta creación incluye veinte temas musicales que nos brindarán casi una hora y cuarto de diversión para nuestros oídos.



Profundizando en el contenido del disco musical y tras varias audiciones, podemos decir que es un tanto especial, por dos razones: la primera es que está compuesto por composiciones totalmente originales interpretadas por los chips sonoros de nuestros MSX, valiéndose del PSG y MSX-Music e intentando retorcer su circuitería para obtener los sonidos más rebuscados y raros, algo común para usuarios activos de la norma, pero a destacar en un recopilatorio musical con perspectivas hacia un reducido mercado musical alternativo, donde se busca un sonido diferente y no necesariamente proveniente de sintetizadores modernos; la segunda razón es que el compacto contiene un tipo de música bastante difícil de catalogar y para nada comercial, pudiendo decir que roza la psicodelia. Personalmente nos parecen más interesantes algunos sonidos, que sin ser tampoco algo extraordinario, superan en conjunto a las melodías en sí. Es decir, no es un disco para amantes de las composiciones tradicionales, como las que encontramos en los juegos MSX, sino todo lo contrario ya que está destinado principalmente a un usuario veterano del MSX que busca experiencias sonoras nuevas y composiciones que no estén forzosamente ligadas con el mundo del videojuego pero sí al de la electrónica, especialmente la de su máquina favorita. En algunos aspectos quizás se pueda encontrar similitudes entre la música de esta recopilación y el estilo de maestros pioneros de la música electrónica como Kraftwerk.

Por último, destacar la excelente calidad de las grabaciones. Presentan una gran nitidez y pureza, poco habitual en este tipo de trabajos procedentes directamente desde el MSX.

El precio de este compacto es de 1995 yen (incluyendo tasas). Si queréis más información acerca de este producto podeis visitar la página de 360º Records (*http://www.360records.net*). También es posible comprarlo, incluso con algún descuento desde otras páginas como Amazon.

Bajan los precios de los cartuchos

Aunque a día de hoy, previo a la Reunión de Usuarios de MSX en Barcelona, no se ha hecho oficial en la página web de MATRA (*http://www.matranet.net*), parece ser que la opción SHOCKWARE va a recibir un suculento descuento.

Poco después de la noticia de la aparición de los nuevos cartuchos por parte de Manuel Pazos, aunque no podemos afirmar que haya relación alguna, Jon Cortázar nos comenta desde los foros de Karoshi (*http://karoshi.msxgamesbox.com*) cómo los precios del servicio que ofrece Matra ha descendido de forma considerable.

Matra ofrece dos posibilidades para los desarrolladores de Software en cartucho. Dicho Soft debe tener formato .ROM y ocupar 8K, 16K, 32K o 48K:

- La primera opción se basa en la opción del pago por cartucho creado, con un mínimo de 10 unidades, al precio de 8 euros por cartucho. Esto incluye cartucho, caja y pegatina. El packaging, a elección del comprador, puede ser diseñado por parte de Matra si se prefiere y cada unidad irá retractilada.

- La segunda opción consigue que el desarrollador no necesite invertir dinero inicialmente. Coincidiendo con las características de formato de la opción 1, se recibirá 2 euros por cada cartucho vendido por parte de Matra. En este caso, el PVP desde matranet será de 10 euros por unidad.

Como podemos observar, hay un trabajo excelente por parte de Matra. Quien no quiera su propio juego en cartucho con noticias como ésta, es porque no quiere.

Lo que se nos viene encima

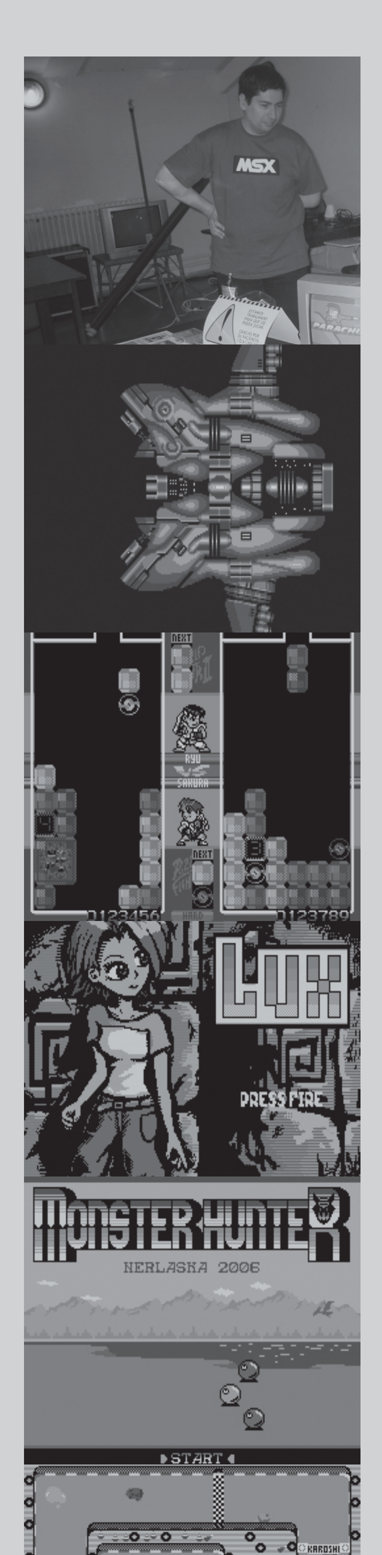
¡Tomad juegos!







16-09-05 **Babaliba** (MSX1 / Karoshi Corporation) - España
Ball-On! (MSX1 / Karoshi Corporation) - España
21-08-06 **Beez** - MSXDEV'06 - (DarkStone) - Holanda
12-03-04 **Blox** (Team Bomba, Graphis 9000) - Holanda
Bust A Move (MSX1 / Karoshi Corporation / Taito) - España
04-02-04 **Carmen Sandiego** (MSX Files / SLotman) - Brasil
13-04-05 **Columns** (MSX1 32kb / Karoshi Corporation) - España
22-04-06 **Cyber Ware: Chapter IV - Katarsis Episode II** (TNI) - Holanda
10-05-06 **Dash!** - (XL2S) - Holanda
04-05-06 **Dragon's Lair** (Nyyrikki) - Finlandia
09-08-03 **Dream On** (MSX2 / Kralizec) - España
26-02-05 **Dr. Mario** (MSX1 / Karoshi Corporation) - España
Dynamite Düx (MSX1 32kb / Karoshi Corporation) - España
30-08-06 **Flash Gordon** (Matra, ROM Cartridge) - España
29-03-04 **Final Race / Final Lap** (MSX1 32kb / Karoshi Corp.) - España
06-01-06 **Goonies 2** (Crappy Soft) - España
16-08-06 **Kame Graphics 2.0** (Paxanga Software) - España
18-04-06 **Krakonia** (MSX1 / Dioniso) - España
Little Knight's Adventure (MSX1 / Karoshi Corporation) - España
19-04-06 **Lucky Darts** (Deltasoft) - Holanda
25-04-06 **Lux** (XL2S) - Holanda
15-05-06 **Malaika-Prehistoric Quest**- MSXDEV'06 - (Karoshi) - España
04-12-05 **Manbow 2** (MSX2 / RenovatiO) - Holanda
02-07-06 **Merin's World** (MSX2 / webmouse) - Holanda
05-12-05 **Monster Bash** (MSX1 32kb, Karoshi Corp.) - España
10-06-06 **Monster Hunter** - MSXDEV'06 - (Nerlaska Studio) - España
03-09-06 **MSX Invaders!** - MSXDEV'06 - (Nerlaska Studio) - España
13-04-06 **Pacman Collection DX** (Op Software) - Holanda
16-08-06 **Pengo** (Paxanga Software) - España
24-06-06 **Project Hunter** (MSX1 / MSX Retrodevelopment) - Holanda
Rapid Burst Space Conflict (MSX1 32kb / Karoshi Corp.) - España
18-06-05 **Soulgrinder** (MSXTurboR / DemonSeed) - Holanda
01-01-04 **Super Puzzle Fighter 2X** (MSX1 / Karoshi Corp.) - España
16-08-06 **Tam Tam Twins** (Paxanga y Dered Software) - España
14-08-06 **Target Area** (Rabbit Soft Workers) - Japón
The Best of Hamaraja Night 2 (XL2S) - Holanda
The Revenge of the Last Dragon (XL2S, Graphics 9000) - Holanda
20-04-06 **TT Virus** - MSXDEV'06 - (Dioniso y WIZ) - España
16-08-06 **Unknown name** (Paxanga Software) - España
26-04-06 **Unknown name** - MSXDEV'06 - (Infinite Software) - Holanda
X-Tazy (Mi-Chi y otros, GFX 9000, Moonsound) - Holanda

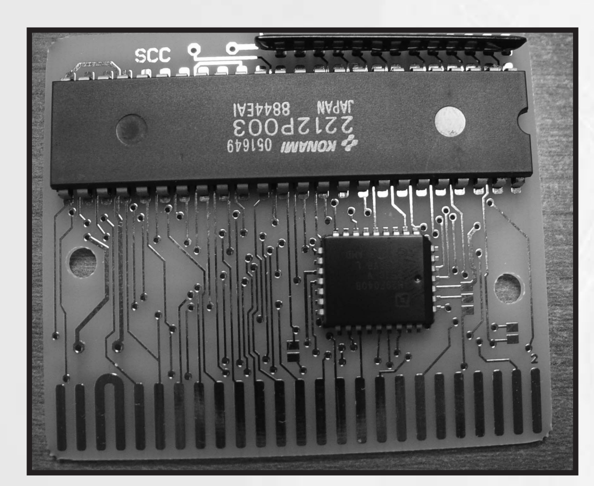


Nuevos cartuchos de Manuel Pazos

Manuel Pazos sorprende durante el mes de Noviembre con par de noticias que no han pasado desapercibidas. Por una parte, ha conseguido crear sus ya famosas MegaFlashROM SCC desde cero, sin necesidad de mutilar un cartucho original de Konami con SCC. Por otra, la construcción de cartuchos ROM para que los desarrolladores de Software tengan un formato físico para sus creaciones.

Tras años de desarrollo silencioso de Software para el estándar, Manuel Pazos vuelve a sorprendernos con una creación Hardware.

Y es que Manuel no descansa. Tras un periodo de estudio y una búsqueda exhaustiva de componentes, ha conseguido crear una placa de cartucho con SCCs tanto para crear las ya conocidas MegaFlashROM como para crear cartuchos ROM.



Y es que hubo alguna gente descontenta con el hecho de tener que “aprovechar” juegos en cartucho para crear las MegaFlashROM. Así pues, este colectivo no tendrá excusa esta vez para adquirirlas.

Además de estas MegaFlashROM SCC de 512K donde cabe hasta un 4 megarom y totalmente compatibles con la original, el bueno de Manuel ha dado un paso adelante y ha creado un novedoso catálogo de cartuchos ROMs para desarrolladores de juegos.

Estos cartuchos soportan juegos o aplicaciones desde 8K hasta 64K sin mapper. Si el desarrollador pensase que esto no es suficiente, tiene la posibilidad de usar mapper para conseguir megaroms de hasta 4 megas de capacidad. Y, para terminar de sorprender a los creadores de Software, éstos podrán hacer uso del chip SCC en sus composiciones musicales.

Sin tiempo para digerir todo lo anunciado y como prueba de potencial del cartucho y oferta de lanzamiento, Manuel ha conseguido reunir diferentes juegos de diversos grupos de desarrolladores -algunos inéditos hasta el momento- y montar un cartucho MultiROM que se venderá por primera vez en la próxima Reunión de Usuarios de MSX al precio simbólico de 10 euros.

Este cartucho, MultiROM Collection, viene con 8 juegos, 3 de ellos no publicados hasta la fecha: Majikazo, nuevo juego de Lemonize del que se sabe más bien poco hasta la fecha; una versión mejorada del Traffic Jam de Imanok y la “Extended Version” del Operation Wolf de Toybox. De estos dos últimos, tenéis una review completa en este mismo número de call MSX.

El resto de juegos que completan el MultiROM son los siguientes: el fantástico The Cure de XL2S, ganador del pasado MSX Dev’05; el peculiar, loco y cachondo Parachuteless Joe de Paxanga Soft; el “invertido” shoot’em up Universe: Unknown de Infinite; el “complejo para el que escribe” Stratos, de CEZ Games Studio; por último, la indiscutible mejor versión bajo MSX que se ha hecho del clásico Tetris: Kralizec’s Tetris.



Logotipo descartado para el cartucho



SymbOS, SISTEMA MULTITAREA

SymbOS es un sistema operativo multitarea que nació en ordenadores Amstrad CPC y que ahora está siendo portado al MSX. Es un programa de origen alemán y su autor es conocido con el apodo de *Prodatron*. En la página del proyecto podemos encontrar multitud de trabajos realizados por el programador, algo que revela el gran nivel de sus conocimientos.

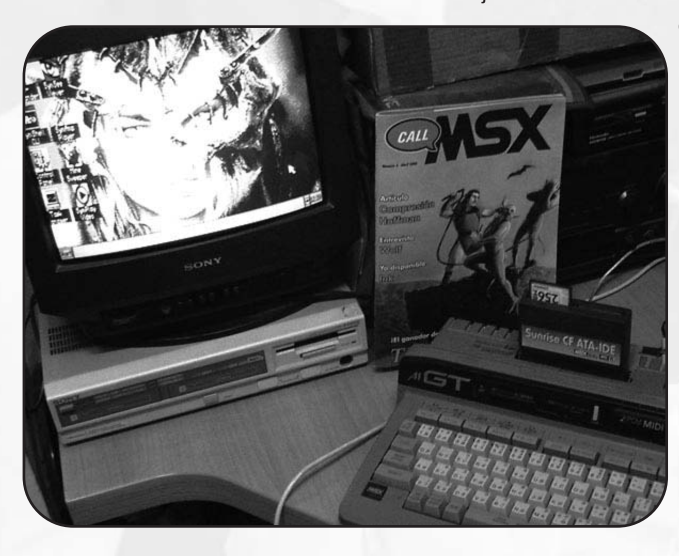
Este sistema operativo tiene una interfaz gráfica similar al conocido Windows, así que disponemos de un escritorio con nuestros accesos directos y un menú desplegable con los programas y las diversas utilidades. Además, alardea de utilizar la multitarea por lo que podemos abrir multitud de ventanas o programas y ejecutarlos a la vez.



Nuestras pruebas

Hemos probado la última versión en un Turbo R con adaptador de tarjetas flash y realmente nos ha impresionado. Podemos escoger una resolución de 512x212 píxels y 4 ó 16 colores, por lo que podemos configurar un aspecto gráfico muy bueno, especialmente si añadimos uno de los espectaculares fondos. El sistema operativo trae un surtido de utilidades entre las que se encuentran una pequeña calculadora, un visor de imágenes (en un formato especial), un reproductor de melodías PSG con aspecto similar al Winamp, un reproductor de vídeo (aunque no lo hemos podido probar) y el clásico buscaminas, entre otras. También tenemos una ventana (*Shell*) para trabajar en modo de comandos, un panel de control e incluso un atractivo monitor de tareas y rendimiento. Existe además una Suite para PC (SymStudio) con la cual podemos crear utilidades, convertir imágenes y vídeos y muchas cosas más para luego utilizarlas dentro de SymbOS.

Multitarea real
576Kb de memoria dinámica
Hasta 128Gb de almacenamiento
100% Interface tipo ventanas



Recientemente, el autor nos ha sorprendido con una versión del clásico Pac-Man. Hemos probado el rendimiento del sistema con varias tareas abiertas y éste ha sido bueno. Con cuatro ventanas de Pacman, el monitor de recursos nos indicaba que se estaba utilizando el 72% de la cpu aproximadamente.

El único punto negativo es para el sistema de instalación ya que es un tanto lioso, engorroso y para nada automatizado. Los problemas han surgido a la hora de ubicar los archivos y establecer las particiones para este sistema operativo. De todos modos, el autor es consciente de esto y está preparando un sistema de instalación más sencillo.

Un aplauso para *Prodatron* por este gran trabajo; esperamos que siga evolucionando, ofreciendo así más posibilidades. ***http://www.symbos.de/***

AVISO

***Para ejecutar SymbOS sobre MSX Turbo R es necesario anular la disquetera del mismo ya que no está todavía soportada, para ello se utiliza el parámetro f0 (hay que escribir symbos f0*).**

29a Reunión de Usuarios de MSX de Barcelona

El pasado 29 de Abril se celebró una vez más la más antigua reunión de MSX que se hace en España. Pese a no ser ya tan populosa como hace una década sigue al pie del cañón. De vez en cuando, si las novedades son llamativas, el número de visitantes se acerca al de los viejos tiempos, pero cuando las novedades son escasas como lo ha sido en esta ocasión, la asistencia puede bajar tranquilamente a la mitad (60 visitantes). El lugar sigue siendo el mismo de las últimas ocasiones, la Sala "L'Altell" en las Cocheras de Sants de Barcelona. Vamos a hacer un breve recorrido por los stands que se presentaron en dicha cita.

El Stand de SEGUNDA MANO

Como viene siendo habitual se hacía cargo de él el presidente de la AAMSX, Jordi Tor. Fue el stand más visitado ya que muchos usuarios buscaban videojuegos antiguos o periféricos. También se vendieron en un suspiro revistas japonesas de MSX Magazine antiguas.



Mr Melenas buscando "cambio"

EL Stand de CALL MSX

Como en cada cita barcelonesa pusimos a la venta un nuevo ejemplar. En este caso el número 6, lo cual significa que llevamos ya tres años en la labor. Uno de los números más equilibrados hasta la fecha en cuanto a contenidos. También vendíamos números antiguos, la trilogía musical *Perfect Covers* de Jordi Tor y unos ejemplares del nuevo *Konami Quiz 2* del grupo Delta Soft que se agotaron pronto. Como siempre llevamos un Turbo R muy bien equipado para mostrar novedades a lo largo de la jornada.



Ejemplares del sexto número de la Call

El stand de PAXANGA

Uno de los grupos que pese a no traer ninguna novedad acabada pusieron stand. Con un cartelito irónico de *En Construcción* dejaban claro que seguían con algún nuevo proyecto entre manos. De paso por si no lo tenías ya podías



Paxanga "under construction"

hacerte con el mítico Yupipati o jugar con el concursable Parachuteless Joe.

El stand de MATRA

Era una incógnita su llegada pero ansiada, ya que todo el mundo sabía que era el día perfecto para presentar el esperado *Ink : Exxon Surfing*, y así fue. Una presentación que resultó curiosa, ya que la mitad de los presentes atendían y un grupo reducido que se situaba en el stand de segunda mano no le hacían ni puto caso. Pues eso, después de muchos años se vendía un cartucho MSX otra vez. Aparte de esto, numerosas camisetas con logos sarcásticos relacionados con la retroinformática o el frikismo televisivo. También se vendían las cintas de *Ink* o *Bloody Paws* aparte de los tradicionales productos MATRA y Kralizec de hace unos años, y números de la revista *First Generation*.



El Mesías quiere pillarse el Ink en el stand de Matra

El stand de TABUROTO

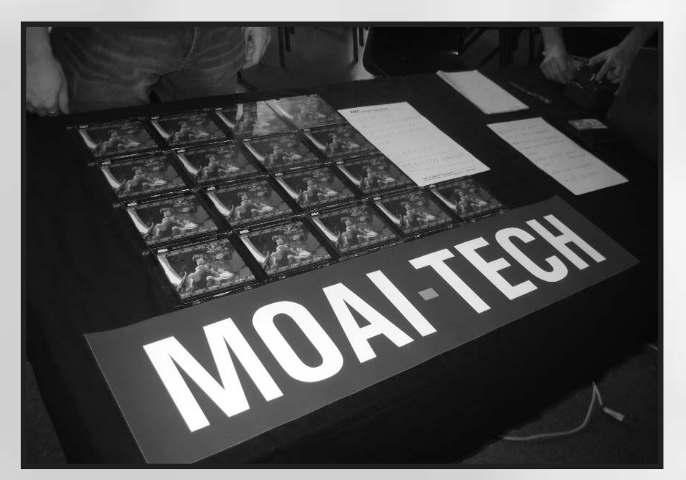
Sin presentar novedad alguna los Taburoto estuvieron liados toqueteando internamente su ordenador y conectándolo al amplificador para poner la música a tope.



Inmerso en una reparación

El Stand de MOAI-TECH

El grupo que presenta periódicamente un webzine vendía un DVD musical llamado *Music Soundtracks eXperience* con 48 discos en su interior en formato MP3 por 2,50 euros. Lástima que hayan optado por este formato tan extendido ya que los puristas en sonido no lo aprecian excesivamente. Pero bueno, si te gusta alguno te lo compras original. También organizaron un concurso con el juego Family Boxing.



Otro Dvd musical de Moai-Tech

Y esto fue todo lo que dio de sí este evento. Por la noche no faltó la típica cena de usuarios para celebrar "Another day of living", aunque esta vez fue más reducida de lo normal.



Una instantánea de la Reunión

¡Inminente lanzamiento!

MANBOW 2

OBLIVION SUN

Manbow 2 es uno de los proyectos más serios y prometedores que podemos encontrar en el panorama del MSX; los creadores, *RenovatiO*, han trabajado duro para que así sea. Hace un tiempo apareció un video promocional en el que podíamos ver las cualidades del futuro juego, entre las que podemos mencionar el suave scroll horizontal y vertical que seguirá fielmente los movimientos de la nave; la velocidad del juego en general y la calidad gráfica. Aunque el planeta que se mostraba en el video era bastante soso, actualmente hemos visto varias capturas de diferentes enemigos y escenarios que nos han dejado con la boca abierta. Un ejemplo es un cangrejo espacial o el mismímo núcleo orgánico que dirigía el Manbow totalmente re-estilizado con unos geniales gráficos (gracias **Norakomi**), el cual vuelve a la carga tras su derrota en la anterior entrega, gobernando de nuevo su poderosa fortaleza.

La nave también ha sufrido la evolución (después de muchas sugerencias y colaboraciones en el foro del MSX Resource Center) y ahora tiene un look más sofisticado.

Respecto al tema musical podemos decir que las melodías son una gozada. Están compuestas utilizando el chip de sonido de Konami (SCC) con la intención de mantener el sonido original del primer Space Manbow. Podemos hacernos una idea descargándonos los tres temas disponibles en la página **bgMSX** (*www.bgmsx.com*).

The Descent y la melodía del enemigo final combinan un sonido electrónico-futurista, ágil, detallista y además melódico, un sonido de la vieja escuela que nos recordará muchas otras legendarias composiciones. *Living Planet* quizás nos recuerde más a la música del original de Konami. De nuevo **Norakomi** lo ha conseguido.

Sin dudarlo le damos un diez a este artista por deleitarnos con este fantástico trabajo, tanto por los gráficos como por la música, aunque estamos ansiosos de ver y escuchar el resto, que si todo va según lo previsto, debería ser presentado a finales de enero en el Encuentro de Nijmegen (Holanda).

El juego

Nada menos que ocho niveles repletos de acción nos aguardarán, a través de los cuales Jr deberá enfrentarse al Manbow espacial y derrotar todo su peligroso ejército.

Nivel 1 - Las cavernas: Jr esquiva el ataque del Manbow pero muchos fragmentos de planeta obstaculizan su escapada. Vuela a través de una caverna que encuentra en un meteorito y allí lucha con multitud de enemigos que permanecían ocultos. Jr derrotará al guardian de la caverna que permanece oculto en las paredes de la misma. Llega la calma y Jr puede pensar en lo sucedido. Vuela hasta las profundidades de la caverna mientras piensa vengar a su padre destruyendo el nuevo Manbow espacial. Finalmente encuentra la salida de la caverna.

Nivel 2 - El espacio: el Manbow espacial detecta la presencia de Jr en el espacio y envía una flota para que lo encuentren y lo destruyan. Jr, lleno de rabia, combate contra estos enemigos y acaba con un gran destructor espacial. La batalla ha sido inevitable pero por el momento ha salido victorioso. Es hora de destruir al gran enemigo.

El Manbow espacial es un sol en estado orgánico que se alimenta de la energía de otros sistemas, destruyéndolos completamente. En su superficie podemos ver un gran ojo, con el que fija su malvada y hambrienta mirada en las profundidades del espacio.

Nivel 3 - Dentro del Manbow espacial: Jr se da cuenta de que su armamento no causa daño alguno en la superficie de esta poderosa nave así que busca una entrada al interior de la misma; encuentra una pequeña entrada y se infiltra a través de ella. Dentro encuentra dificultades para esquivar a la gran cantidad de enemigos pero logra llegar al corazón. El sistema de defensa de éste se activa mientras Jr intenta destruir el corazón si éxito; la batalla se hace cada vez más difícil. Un rayo sale del corazón del Manbow y abre un agujero hacia el exterior; esta vez, Jr no tiene más remedio que escapar.

Jr piensa cómo destruir a este poderoso enemigo, así que pone rumbo hacia las ruinas Alpha4 con el fin de encontrar la

solución.



Nivel 4 - El descenso: viejos templos, columnas destrozadas y restos de lo que antiguamente fueron gigantescas estatuas componen el paisaje de este planeta. Jr lucha contra multitud de estrañas criaturas a pesar de no ser unos enemigos prioritarios para él. Finalmente se enfrenta contra una antigua e inmensa nave espacial. Después de la lucha busca alguna pista entre las ruinas pero no encuentra nada. Una extraña señal proveniente de un cercano planeta hace que Jr ponga rumbo hacia él.

Nivel 5 - El planeta viviente: un planeta dominado por el agua y cubierto por una gran jungla floral. Hay un pequeño pueblo y Jr desciende hasta él. Los habitantes entienden la causa del piloto y deciden ayudarle, pero a cambio piden que Jr libere al pueblo de un horrible monstruo que cada día ataca al poblado. Pilotando su nave a través de la frondosa selva encuentra la guarida del feroz enemigo: se trata de un dinosaurio. La batalla es frenética pero finalmente el gigantesco animal es derrotado.

Jr es proclamado el héroe del pueblo y descubre así que su gente son descendientes directos de la civilización Alpba4. Cuentan cómo sus ancestros consiguieron grandes avances tecnológicos aprendiendo a sustraer la vida de los ecosistemas, pudiendo crear formas de vida completamente nuevas. Desafortunadamente, su sol fue prácticamente destruido en una colisión contra un gran meteorito, así que desarrollaron un plan para enviar energía vital al astro. Al principio la idea funcionó pero al poco tiempo este nuevo sol tomó consciencia propia y aprendió a robar energía de los ecosistemas, así que comenzó a destruir todos los planetas cercanos. Como esto podría suponer la destrucción del Universo se diseñó un arma láser para destruir este mortal sol llamado el Manbow espacial.

El Manbow fue destruido. La gente del planeta lo abandonó y se dirigió al planeta donde Jr se encuentra. Los científicos que diseñaron el láser se dirigieron a un lugar desconocido del espacio con la idea de revivir el Manbow espacial. Ellos todavía mantienen el increíble arma láser.

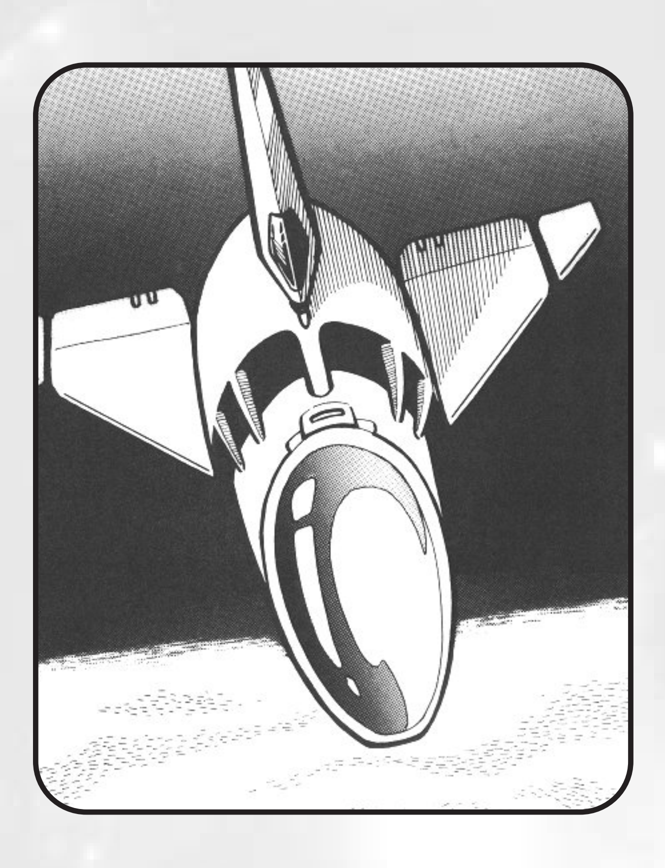
El pueblo todavía tiene la bomba que puede destruir el escudo que proteje el corazón del Manbow, pudiendo así destruirlo con la ancestral arma láser.

Jr recoge la bomba y comienza la búsqueda del láser.

Nivel 6 - Profundidades del núcleo: Jr entra en una atmósfera decadente. La muerte domina la superficie y restos de anterior vida se espacen por ella creando una manta oscura de ceniza. Solamente los desalmados y la maldad puede vivir en este ecosistema. Jr lucha contra multitud de demonios y llega hasta un oscuro hangar donde se enfrenta a una gran nave orgánica. Finalmente logra entrar dentro del complejo y encuentra el láser. Es hora de la venganza.

Nivel 7 - Rojo / Amarillo: el Manbow espacial envía nuevamente un escuadrón para evitar que Jr se aproxime. Jr acaba fácilmente con el escuadrón y su líder.

Nivel 8 - Dentro del Manbow espacial: La lucha será espectacular y decisiva para el futuro del Universo. ¿Conseguirá Jr destruir al Manbow espacial y salir con vida? Haceros con este fantástico juego para conocer el desenlace.



Ya disponible



1986, Konami daba al MSX la primera entrega de una exitosa saga, Mayou Densentsu, más conocido en Europa por Knightmare. Han pasado los años y siempre ha estado ahí como referente, un juego que nunca pasará de moda, un clásico donde se inspirarían juegos posteriores. La propia Konami le añadió en 1990 sonido SCC+ que mejoraba notablemente sus famosas melodías. Se han hecho remakes, se ha llevado a móviles y ahora tenemos una nueva revisión justo cuando cumple 20 años de su primera aparición.

Lo primero que nos pasa por la cabeza antes de probar el juego es si realmente valía la pena retocar un juego que ya nos parecía "intocable", se había convertido en clásico por méritos propios. Aunque esta pregunta no es una novedad porque hemos visto ya en otros casos cómo se han "retocado" juegos antiguos con un magnífico resultado. Es el caso de *La Abadía del crimen*, donde Manuel Pazos hizo un excelente trabajo y le dio el aspecto cromático que merecía nuestro sistema.

En este caso los responsables han sido los programadores del grupo brasileño Amusement Factory. El hardware requerido para poder jugar con él es variable, todo depende del modo al que quieras jugar; vamos, que se hacen un buen número de combinaciones según el modelo de ordenador que dispongamos aparte de los periféricos. Lo mínimo es un MSX1 con el que jugaremos a la versión original, pero si disponemos de dispostivo para CDRom funcionarán las pistas de audio.

Como siempre en esta vida lo mejor es disponer de un Turbo R para curarse en salud y la unidad de CDRom con la controladora ATA-Ide. De este modo nos funcionará el juego a la perfección.

Y es que nunca habíamos jugado en MSX con música en CD, se nos hace rarísmo. Lo más parecido a un MSX con CD es la mítica Pc Engine Turbo Duo. Pero bueno, ya tocaba. Ya que teníamos la posibilidad de disfrutarlo aquí lo tenemos.

Lo que no tiene sentido en este comentario es hablar del juego en sí, es de sobra conocido, así que hablaremos de las novedades que presenta.

Comenzamos por el apartado gráfico. Si alguna vez habías soñado cómo podría ser el Knightmare para MSX2 ya puedes abrir los ojos y despertar. El colorido es excepcional. La versión clásica ya disponía de un color acertado pero esta vez se ha sabido matizar con elegancia las nuevas posibilidades de la paleta de los ordenadores de segunda generación. Se ha



optado por elegir una gama más "apastelada" que da una sensación de dulzura cromática. Los colores han perdido intensidad pero están más armonizados. De este modo destaca más el dibujo porque el negro sí que mantiene su fuerza intensa.

Se ha retocado el scroll, si lo pruebas en un ordenador que supere los 7 mhz vas a notar la diferencia. Es muy suave, realmente ha ganado en jugabilidad, y eso ya era difícil. Pues ya desde un principio percibes una sensación diferente, muy agradable en el movimiento del personaje. Recordar que el original comenzaba como demasiado lento, parece que le costaba moverse. Hasta que no cogías un par de velocidades no lo encontrabas satisfactorio. Ahora desde buen principio notamos esta sensación.

El sonido es el otro punto fuerte de esta revisión. La gran novedad reside en las pistas de audio, una sensación desconocida hasta ahora en el MSX y que tendrá tanto detractores como gente a favor. La banda sonora respeta la original y añade nuevos temas familiares de la casa Konami. Los arreglos son más que correctos. Los efectos de sonido se mantienen en PSG. En este caso debemos ajustar el nivel de los altavoces con el nivel del volumen del televisor para poder equiparar el sonido. Si ponemos muy alto los altavoces en comparación a la TV no oiremos los efectos. Es en definitiva lo mismo que sucede con el Moonsound y el PSG.

Knightmare Gold

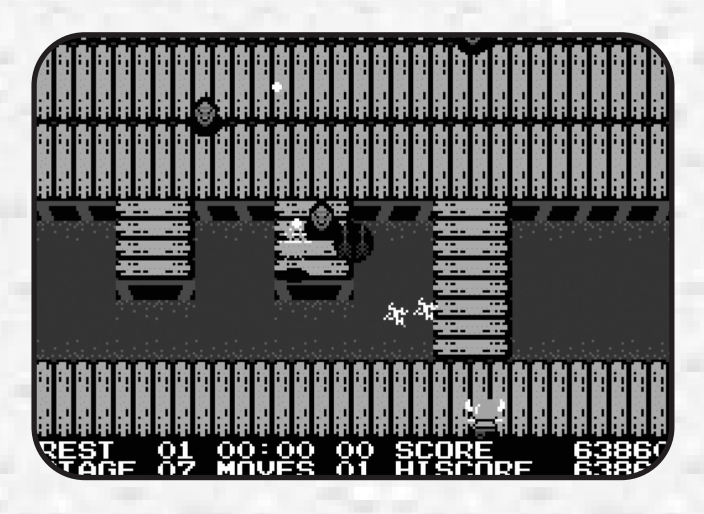


Puede surgir un pequeño problema según el lector de CD que dispongamos. Con lectores actuales no hay problema pero si tenemos algún modelo bastante antiguo de aquéllos de 4 velocidades sí vamos a notar unos instantes de parón (pantalla negra durante 1 incordioso segundo) mientras carga la pista de audio del cd. Hemos probado con varios lectores y con los de mayor velocidad se juega en perfectas condiciones. En cambio con el lector lento se hace insoportable a no ser que quites los efectos de sonido del CD por los del PSG, de este modo la cosa se hace jugable.

Si vemos algún otro "fallo" es ya propio de la ROM original. Nos referimos a cuando en algún caso un enemigo pasa como Jesucristo por encima de las aguas, en lugar de por el puente. Como decimos esto ya sucedía en la ROM original, así que nada de error.

La lástima es que no se hubiera conservado el modo de sonido con chip SCC+ de la propia revisión de Konami por si no eres partidario de las pistas de audio. No sabemos si algún día será implantado por petición popular, pero parece que no estaban por la labor los programadores.

Y quitando las posibles "pegas" también debemos celebrar



las ventajas. La principal es que se ha conseguido que dejen de parpadear algunos sprites. La razón, muy sencilla. Al ser reprogramado como un juego de MSX2, la nueva máquina concede más sprites por línea lo que han sabido aprovechar los programadores para mejorar este "defecto".

En conclusión, estamos ante un "patch" del juego original, no un remake. No han tratado de sustituir el juego sino de añadirle extras, y creemos sinceramente que ha valido la pena por abrir nuevos campos sin explorar en el MSX como son las pistas de audio. Todo lo que se ha tocado ha sido con la mejor intención de hacer de un juego extraordinario un poco mejor si cabe. Debemos felicitar esta iniciativa y esperar a que se atrevan con nuevos proyectos para el futuro.

Jugabilidad:	9
Música:	9
SFX:	7
GFX:	8
Duración:	8
Extras:	9
Total:	**9**



Celebramos proyectos como éste donde se puede mejorar lo existente. En nuestro sistema se pueden mejorar muchísimas cosas debido a los nuevos accesorios existentes. No sólo las mejoras de color y audio son bienvenidas. Algo que no estaría mal pero dudamos de que se pueda llevar a cabo es en los juegos de disco y sus molestas cargas de 64k de RAM. Se debería conseguir que los numerosos discos fueran un único archivo para las nuevas potentes tarjetitas de memoria. De esta forma no se cortarían las melodías.

¡CREA TU CD KNIGHTMARE GOLD!

Para empezar debemos dirigirnos a la página oficial de Knightmare Gold que se encuentra en siguiente enlace: *http://www.caetano.eng.br/MSXPage/KMG/*

Dentro de la sección "Arquivos" hay que descargar todos los archivos del programa (*pacote principal* y los archivos MP3), en total 22 ficheros. La descarga es un poco lenta por lo que hay que tener un poco de paciencia. Opcionalmente también podemos descargar los manuales y la portada, así tendremos el juego como si de un original se tratara.

Una vez que tengamos todos los archivos en nuestro PC, descomprimimos el archivo *KMG_R2.ZIP* en *C:\KMG\GAMEDATA*. Seguidamente tenemos que buscar la ROM original de Knightmare (que no se incluye en este pack) y copiarla con el nombre de *KMARE.ROM* al directorio anterior. En total deben quedar cuatro archivos en este directorio.

Seguidamente debemos convertir todos los ficheros MP3 a WAV, para ello podemos usar cualquier programa que pueda hacer la conversión. Nosotros hemos utilizado el *Nero Wave Editor*. Los ficheros convertidos deben mantener el mismo nombre con la nueva extensión WAV. Cuando tengamos los



Cd de Konami Gold preparado para su lectura

21 ficheros de audio, hay que copiarlos todos en una carpeta que crearemos con el nombre de *C:\KMG\GAMEISO*

Ahora hay que preparar la ISO de los datos que serán grabados en el CD junto con las pistas musicales; para ello utilizamos el programa *MKISOFS*, para la versión de sistema operativo que utilicéis; nosotros explicaremos cómo hacer todo el proceso para Windows. Esta utilidad se puede descargar de muchas páginas de Internet pero recomendamos hacerlo desde *http://www.sbox.tugraz.at/home/t/tplank/* puesto que hemos descargado otras versiones de estas utilidades que han dado errores y "cuelgues". En esta página encontraréis un paquete de utilidades que se llama ***cdrtools-2.01.01a17-win32-bin.zip*** que contiene las utilidades necesarias para la creación del CD, también es imprescindible bajar la DLL ***cygwin1.dll*** para el correcto funcionamiento de éstas. Si copiamos los archivos ***cdrecord.exe***, ***mkisofs.exe*** y ***cywin1.dll*** dentro de la carpeta de windows, no deberíamos tener proble-







Running the Game

mas. Una vez preparado todo esto, procedemos a crear la imagen de datos: vamos a *Inicio->ejecutar* y escribimos *CMD* para entrar en modo de consola de comandos. Escribimos:

C:
CD KMG
MKISOFS -V KMG_R2 -iso-level 3 -RJ -o .\gameiso\k01.iso .\gamedata

Si hemos seguido todos los pasos correctamente deberíamos tener los siguientes archivos en el directorio *C:\KMG\GAMEISO*:

K01.ISO
K02.WAV
K03.WAV
K04.WAV
K05.WAV
K06.WAV
K07.WAV
K08.WAV
K09.WAV
K10.WAV
K11.WAV
K12.WAV
K13.WAV
K14.WAV
K15.WAV
K16.WAV
K17.WAV
K18.WAV
K19.WAV
K20.WAV
K21.WAV
K22.WAV

Ahora sólo queda grabar la información sobre un CD.



Dispositivo IDE necesario para conectar el lector de CD

Desde la consola de comandos escribimos:

C:
CD KMG\GAMEISO
CDRECORD -SCANBUS | MORE



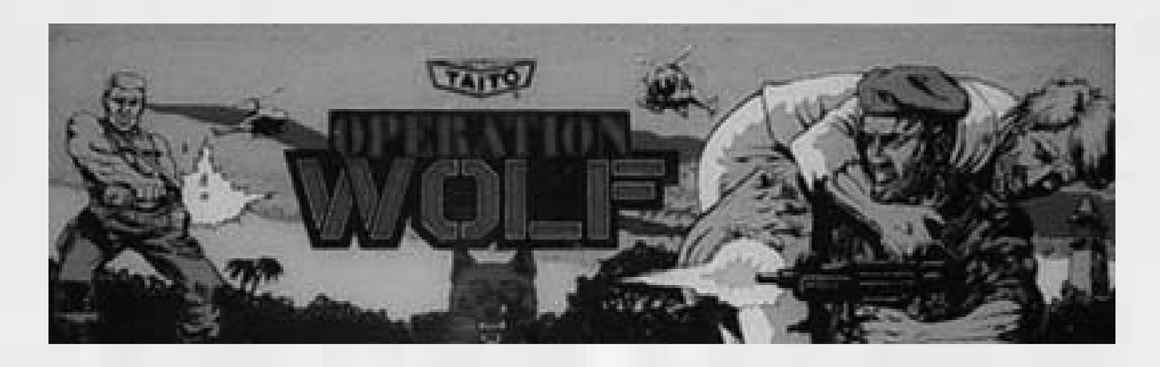
Knightmare Gold funcionando en un TurboR

Obtenemos la información de nuestra grabadora de CD/DVD. Hay que apuntar los tres valores separados por comas asociados a dicha grabadora. Colocamos un CD virgen y escribimos:

CDRECORD dev=x,y,z speed=4 fs=8192k -v -pad -multi -xa k01.iso k02.wav k03.wav k04.wav k05.wav k06.wav k07.wav k08.wav k09.wav k10.wav k11.wav k12.wav k13.wav k14.wav k15.wav k16.wav k17.wav k18.wav k19.wav k20.wav k21.wav k22.wav

Donde *x,y,z* son los valores de nuestro grabador, obtenidos al principio. Una vez finalizada la copia y si no aparece ningún error, el juego estará listo para ser disfrutado.

Hemos realizado las pruebas de juego sobre un Turbo R GT con un adaptador Sunride ATA-IDE y un lector DVD Toshiba de 16x y ha funcionado todo a la perfección. Recuerda que para que funcione la música el juego debe indicar que ha detectado el CD (*KMG CD found*), comprueba bien la configuarción del IDE y asegúrate de ejecutar el driver de CD (*IDECDEX*).



¡VUELVE OPERATION WOLF!

El equipo de TOYBOX lo ha conseguido. Una versión del clásico de Taito, digna de nuestras máquinas, en sólo 32 kbytes de memoria (versión MSXDEV'06) y 48 kbytes para la versión extendida, con soporte para MSX-Music y compatible con todos los ordenadores y generaciones de la norma. Ya podemos disfrutar de esta pequeña joya que luchará contra el resto de proyectos del concurso y que a pesar de su gran calidad parece ser que no lo tendrá fácil.

Muchos recordaréis el juego original de 1987 creado por Taito en el que, a través de una vista subjetiva en primera persona, debíamos apuntar con nuestra ametralladora (físicamente presente) y disparar a todos los enemigos que encontrábamos a nuestro paso; una idea por aquella época bastante original y que hizo que el juego tuviera mucho éxito.

En 1988 Ocean adaptó, como de costumbre, la versión de Spectrum directamente al MSX, obteniendo un juego lento y carente de buen colorido; sin embargo y a pesar de todas estas pegas, la conversión todavía era jugable y podías pasar ratos divertidos, incluso con las interminables cargas desde la cinta de cassette. De todos modos, no aprovechaba para nada las prestaciones del ordenador japonés.

Así que para variar, este fue un juego versionado para todas las máquinas existentes. En un par de años el juego estaba desde el Spectrum hasta el Amiga, pasando por las consolas propias de aquellos años, Nintendo, Master System y Pc Engine.

Después de nada menos que dieciocho años desde todo aquello, aparece la versión que siempre se mereció nuestro sistema, llena de colorido, rapidez; una versión realmente difícil y adictiva. Multitud de sprites se mueven suavemente por la pantalla y a pesar de que se han suprimido los primeros planos de enemigos podemos asegurar que mantiene una calidad gráfica excelente y un rediseño más ajustado y equilibrado para nuestros MSX.

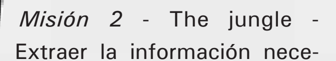


La versión extendida recupera los gráficos en las intros

Los 16KB extras de la versión extendida no se quedan en una mera adición de músicas o más pantallas como suele ser habitual en otros juegos, aprovechando el mayor tamaño de la ROM. Aquí se ha sabido dotar al juego de lo que realmente carecía, aunque sean añadidos que no modifiquen el juego en sí (salvo por el uso de la paleta MSX2). El resto hace que un juego que podría parecernos "incompleto" por no contener los extras que sí aparecían en la recreativa original consiga una puntuación global muy alta y el estatus de videojuego con un acabado muy profesional en la parte software.

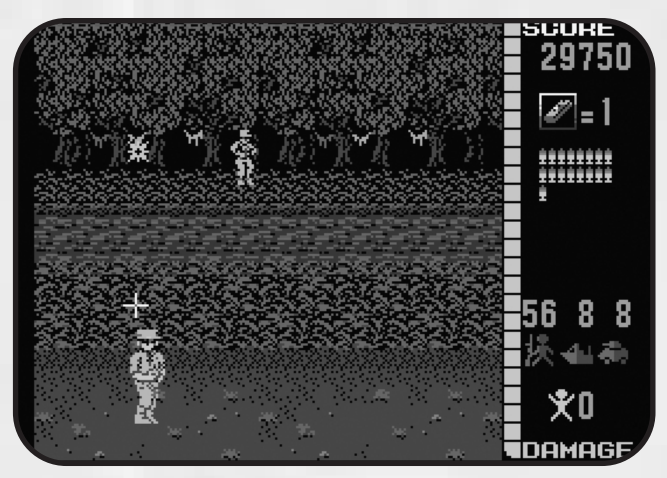
-El juego se divide en seis misiones.

Misión 1 - Communication Setup - Destruir el sistema de comunicaciones enemigo y evitar que avancen sus tropas. Un ataque directo a la base enemiga donde se encuentran multitud de soldados, vehículos blindados e incluso helicópteros.



Misión 2 - The jungle - Extraer la información nece-

saria para localizar el campo de concentración. Nos movemos siguiendo el curso de un río que atraviesa una frondosa selva.



La segunda misión nos llevará a la jungla

Misión 3 - Village - Libera la aldea de la ocupación enemiga y toma un merecido descanso. Dentro de la aldea la concentración es máxima, debemos tener especial cuidado en no disparar a los civiles que cruzan.

Misión 4 - Powder Magazine - Dentro del polvorín debes vigilar con los enemigos que se esconden tras los búnkers y trata de recuperar toda la munición posible.

Misión 5 - Concentration Camp - Tienes que ayudar a los cinco rehenes y ponerlos a salvo. Si consigues salir de ésta huirás con los ellos.

Misión 6 - Airport - Debes conducir a los cinco rehenes hasta el avión de rescate y ejecutar una precipitada huída. Debes salvar todos los rehenes para salir victorioso.

Resumiendo, por fin tenemos un Operation Wolf como se merecía el MSX en aquella época que mejora la burda conversión del Spectrum. Puestos a comparar, Toybox ha creado



También se incorpora el mapa existente en otras versiones

una versión de las más dignas vistas en 8 bits, sobretodo con la incorporación de los extras (intros, mapa, control vía ratón y selección de 60 hz). Tanto la versión de NES como la de Master System carecen de enemigos en primer plano, así que no podemos criticar a Toybox por la supresión de éstos.

La música de esta nueva versión supera en calidad a ambas, incluso a la de Pc Engine. Suena pero que muy bien. Se ha acertado con crear las melodías para el chip Msx Music. Tampoco notamos nada extraño en los efectos de sonido, lo cual quiere decir que cumplen con su cometido. Disparos normales, ametralladoras, explosiones o el efecto de cuando te lanzan una granada están conseguidos.

El apartado gráfico es correcto pero sin alardes. Todo son sprites, de dos colores los delanteros y monocromos los posteriores, y les falta volumen, pero es preferible eso a la versión incolora de hace 2 décadas. Además, la pantalla de juego propiamente dicha dispone de mayor tamaño respecto a la antigua. Si la comparamos con las versiones consoleras aquí salimos perdiendo. Debemos reconocer que la Master System es una pedazo de máquina en potencia gráfica y es la mejor versión visualmente hablando.

OTRAS VERSIONES DE OPERATION WOLF



Arcade

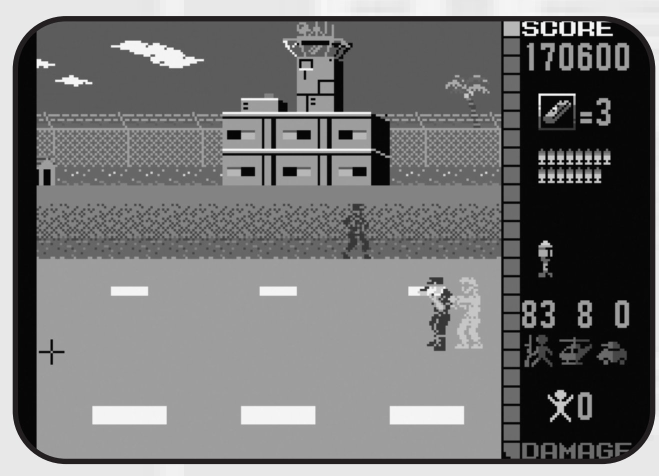


Pc-Engine



NES

Entre las "pegas" gráficas podemos decir que en la segunda fase, en la jungla, cuesta ver los ítems que caen a nuestros pies por la similitud de color de éstos con el del terreno. Es decir, bombas verdes en zona de vegetación, por ejemplo. Y también sucede algo similar pero más vistoso en la última pantalla, en el aeropuerto. Este "fallo" lo localizaremos si hemos mantenido pulsado Graph durante la ejecución de juego para seleccionar la paleta MSX1. El color tan luminoso del asfalto dificulta la visión de los sprites y complica la labor de acertar a primer golpe de vista entre enemigos y rehenes. Hay que decir que en MSX2 esto no sucede. Esto es debido a que se ha programado en MSX2 y al trasladar los colores a la paleta de MSX1 han aparecido sorpresas como ésta. Es un problema similar al que ya vimos hace años en Scramble Formation, no en el error de paletas, pero sí en la confusión cromática, recordemos que era difícil ver los sprites de los disparos enemigos.



Los rehenes deben llegar sanos al avión

Los extras de la versión extendida son de agradecer, se pedían a gritos. Los gráficos de la intro o del game over son exactamente los mismos que en el arcade o las otras versiones pero adaptados al MSX. No se trata de un simple ripeado, o por lo menos han sabido camuflarlo adaptando los gráficos y los colores al formato MSXero. También el mapa de las fases ha sido incorporado de manera correcta, con gráficos que recrean las diferentes fases además de indicar el número de enemigos que debemos derrotar, tanto soldados de a pie como helicópteros y tanques.



¡Y es que quedarse sin munición pasa factura!

La jugabilidad ha ganado enteros con la inclusión del ratón. ¡Es una auténtica pasada! La mirilla va de una punta a otra de la pantalla a gran velocidad y con una precisión envidiable. Digamos que gracias al ratón tenemos posibilidas reales de acabar el juego. Ya no sólo serán capaces los usuarios de más pericia con los cursores (o joystick) de llegar al final sino que cualquier usuario con un mínimo de atención podrá llegar muy lejos. Esto motiva a más usuarios a engancharse, ya que un juego realmente difícil quita las ganas de seguir jugando. Y esto no es sólo para el modo MSX2, sino que las rutinas del ratón han sido creadas para MSX1, ya que la primera generación del estándar no disponía de soporte en la BIOS... ¿recordáis software para MSX1 que soportara ratón? Para terminar de hablar de la genialidad del uso del ratón cabe comentar que, tanto él como el joystick, funcionan en ambos ports. Es más, pueden conectarse durante las interfases, que serán detectados y completamente funcionales.

En fin, podrás conseguir Operation Wolf versión extendida, entre otros juegos, el 9 de Diciembre de 2006 en la Reunión de Usuarios de MSX de Barcelona dentro de un cartucho multirom presentado por Manuel Pazos. Vale la pena hacerse con él porque creemos que los extras que conlleva son más que suficientes como para no quedarse únicamente con la versión limitada del concurso.

Este Operation Wolf es un juego que, sin duda, engrandece todavía más una idea absolutamente genial de ofrecernos en cartucho estas pequeñas joyas...

Jugabilidad:	9
Música:	9
SFX:	8
GFX:	8
Duración:	8
Extras:	9
Total:	**8**

traffic jam



Hacía tiempo que Imanok no nos presentaba ningún programa, el último fue el Cat'n Mouse. En esta ocasión, el colomense nos sorprende abandonando su habitual programación en TurboBasic y se mete de lleno en el ensamblador. Resultado, Traffic Jam, un juego que se presentará dentro del cartucho multirom que Manuel Pazos pondrá a la venta en la próxima Reunión de Barcelona. A la par participará en el concurso MSX-Dev'06 con el mismo título, pero acotado en prestaciones: En MSX2 no permite simular la paleta MSX1 arrancando con GRAPH, las melodías sólo usan PSG cuando la versión que trae el Multirom posee FM y tiene un total de 20 pantallas en vez de 40.

Podríamos clasificar a Traffic Jam conceptualmente entre el rompecabezas de toda la vida y el mítico Sokoban. Debemos superar el puzzle poniendo orden en el caos. La excusa es conseguir que Sandy Olson, una mujer policía, saque el coche de su amigo Danny Zucku del atasco circulatorio que se monta a diario. Para ello debemos dirigir el movimiento de todos los vehículos colapsados para facilitar la salida del atasco del coche de Danny. Los vehículos hay que tomarlos como piezas del rompecabezas, las hay más cortas y más largas, para hacer más complicada la maniobra. La dificultad irá subiendo a medida que vayamos superando fases. Cada cinco obtendremos un merecido password, habitual ya en títulos de Imanok como el Bubble Bobble.



Gráficamente el programa es muy elemental. No es necesario ningún tipo de alarde en un juego como éste. La pantalla de presentación es muy básica pero respira al no tener exceso de elementos con una tipografía más que correcta. La pantalla de juego se desarrolla dentro de una pequeña ventana donde se sitúan los vehículos que tienen un nivel de detalle muy logrado (apetece un juego de conducción con estos coches) . En la parte derecha tenemos un marcador donde nos indica los movimientos realizados durante la maniobra y el tiempo restante. En la parte inferior, Sandy Olson nos irá poniendo al corriente de la situación diaria del tráfico. El dibujo es nítido pero, desde nuestro punto de vista, se debería evitar caer siempre en grafismos típicos del manga, aunque esto ya se sabe que es muy subjetivo.

No hay mucho que comentar sobre la jugabilidad, tan sólo debemos seleccionar el vehículo a mover con el icono del dedo y manteniendo la barra espaciadora movemos el vehículo en la dirección posible. Los cursores responden suave y hacen que el juego sea dinámico y agradable, con lo que el vicio está asegurado.

La melodía utiliza chip MSX-Music, cosa que se agradece en juegos de MSX1, ya que en otros tiempos era algo poco común. Por suerte, se ha escogido con mimo, pues tras horas de juego no agobia en absoluto.

En resumen, nos alegramos por la vuelta de Imanok y esperamos que con sus nuevos conocimientos en ensamblador logre desarrollar juegos aún mejores que los que nos tiene acostumbrados. Desde aquí le animamos para que el próximo sea un título más elaborado y demuestre su potencial. Éste, sin duda, sienta las bases de una forma muy digna.

Jugabilidad:	8
Música:	7
SFX:	6
GFX:	7
Duración:	8
Extras:	6
Total:	**7**

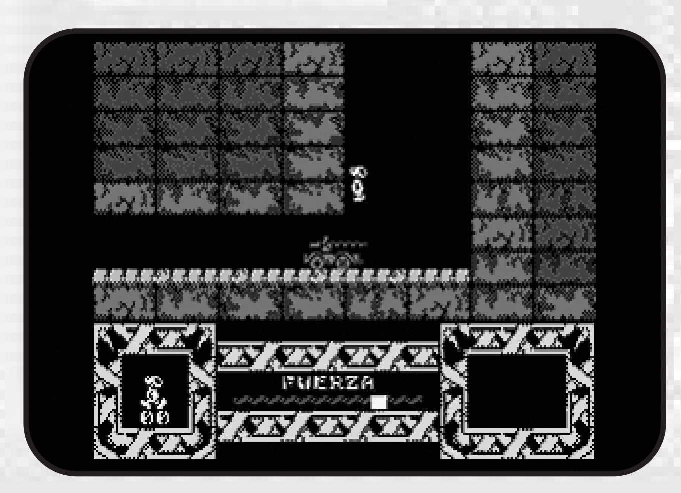
Ya disponible



Phantomas' saga Infinity era un juego programado por el grupo CEZ Games Studio para los sistemas Spectrum y Amstrad. Ahora gracias a la labor de Jon Cortázar (Karoshi Corp) tenemos una versión para MSX1 en formato Rom y con los gráficos redefinidos aprovechando la mayor capacidad gráfica de nuestro estándar.

La salida de este juego nos ha pillado en plena maquetación de la revista. Aunque estábamos pendientes de su desarrollo no esperábamos por estas fechas su lanzamiento. No es el primer juego que este grupo saca para MSX. Ya participaron en el concurso MSXDEV'04 con su programa *Stratos*, aunque no es un grupo programador exclusivo para MSX, ya que trabajan tanto para Spectrum como para Amstrad.

Antes de nada hay que saber de dónde viene la idea en la concepción de este juego. La mítica firma española Dinamic realizó dos juegos basados en la saga Phantomas. Tan sólo la segunda parte, *Vampire* (1986) fue producida para nuestro sistema, pero era un calco de la versión de Spectrum, así las posibilidades de mejora en MSX se quedaban al margen.



Captura de Phantomas 2 Vampire de Dinamic

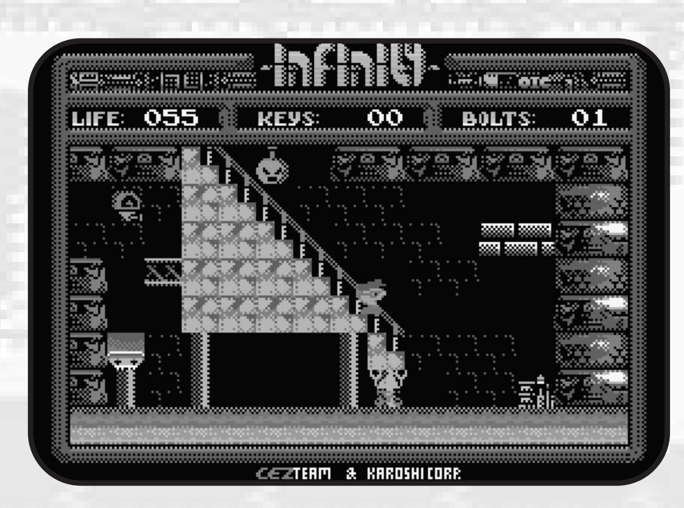
En esta ocasión el grupo CEZ Games Studio homenajea esta saga realizando un juego totalmente nuevo basado en el mismo personaje, algo parecido a lo que ha hecho RenovatiO con Manbow 2. La historia es bastante compleja, pero el objetivo es claro, destruir la factoría de Androides saqueadores. Para ello debemos activar diez dispositivos encontrando para cada uno su llave. Una vez hayamos activado las llaves en sus cerrojos correspondientes, habremos activado los diez dispositivos. Lo cual significa que la bomba está a punto de estallar. Así que tienes medio minuto para abandonar la factoría, o lo que es lo mismo, volver a la pantalla inicial. Esta angustiosa huida contrarreloj ya los habíamos vivido en Metal Gear o la saga Metroid de Nintendo, por poner un ejemplo.



En la pantalla de presentación ya percibimos varias sensaciones. En primer lugar el grafismo, hacía un par de décadas que no veíamos algo similar. Gráficos típicos de las conversiones de Spectrum con la técnica del puntillismo, para lograr ciertos degradados. Eso sí, muy colorista, por eso nos sorprende. Estábamos acostumbrados a relacionar esa estética con la monocromía, salvo raras excepciones, como el Magical Kid Wiz, pero claro, éste era japonés. El otro recurso que nos llama la atención es el scroll vertical de fondo mientras tiene el logo por delante. Una técnica que ya vimos en su momento en el Tetris de Kralizec y que lo hace muy vistoso, y consigue dar un toque de dinamismo a las siempre estáticas pantallas de presentación.

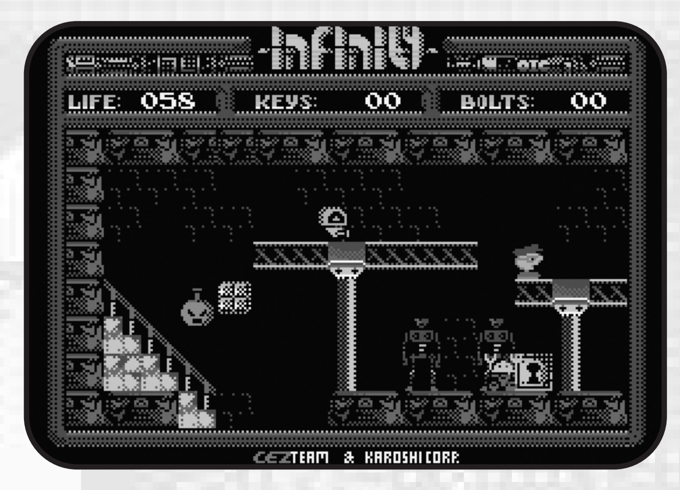
El juego en sí sigue la estela de muchas producciones ochenteras del estilo plataformero.¿A quién no le viene a la cabeza el Jet Set Willy o el Abu Simbel Profanation? Nuestros

movimientos son limitados, derecha, izquierda, salto alto pero corto y salto largo pero bajo. Costará hacerse con el control si estamos acostumbrados a subir y bajar por escaleras. Aquí eso no vale. El triunfar en esta misión depende de cómo dominemos los saltos, el adecuado para cada momento. Sólo tenemos una vida que irá desvaneciéndose a medida que vayamos topando con los enemigos. Conseguiremos algunos puntos de vida adicional si logramos abrir un cerrojo. Hay que reconocer que el juego es bastante difícil, ya que sortear a algunos enemigos no es pan comido. Además cuando cambiamos de pantalla no sabemos con quién nos toparemos. Es fácil caer encima de un enemigo si saltamos de una pantalla a otra. En este caso jugabilidad y dificultad podrían ir estrechamente relacionadas. Porque estamos ante una jugabilidad de precisión. Un movimiento fatal te hará perder bastante energía, así que hay que tener especial cuidado. El problema es que se requiere precisión en momentos donde tienes el enemigo encima y acabarás dándole cuatro puñetazos al teclado de pura rabia.



Los gráficos del juego son correctos. Si lo que te gustan son los gráficos puliditos y definidos olvídate. Estamos ante un grafismo un poco tosco pero que se defiende muy bien a base de una riqueza en cuanto a texturas. Si se saben controlar el resultado puede ser muy decente. Quizás le falte algo de carisma a los personajes. Como punto a favor decir que al tratarse de una conversión de otros sistemas de ocho bits el esfuerzo por redefinir los gráficos para su optimización en MSX es de agradecer. Es la mejor versión gráfica de todas. Usuarios de Spectrum y CPC se podrán dar cuenta por fin que el MSX siempre fue infravalorado en las conversiones, que era una máquina a la que se le podía sacar mayor partido.

El aspecto sonoro está en la línea de este tipo de juegos. La música de la presentación sigue los patrones de las melodías de los juegos españoles de los años ochenta, muy típica de los ordenadores Spectrum, forzando en ocasiones el bajo para que ofrezca una sensación de ruido. La melodía del juego está muy lograda, aunque nos la imaginamos más en un juego espacial. Podría pasar como un tema más en el Zanac o en el Alpha Roid. Pero está muy bien, aunque no sea un tema que se te quedará nunca grabado en el cerebro consigue una atmósfera muy especial. El juego tiene dos temas más pero hay que llegarse al final para poder oírlos, los cuales están realmente bien. El sonido FX cumple perfectamente su cometido. Tanto los saltos, impactos, el coger las llaves etc. son más que correctos. Se echa de menos algún efecto sonoro en el caminar del personaje, algunos pequeños sonidos para simular los pasos, le darían más consistencia, más peso a un sprite que ya de por sí le falta algo de volumen.



En definitiva, un nuevo título al catálogo de nuestro sistema que no hace otra cosa sino que demostrar el gran nivel de profesionalidad de los programadores actuales, que no tienen nada que envidiar a los que se ganaban la vida hace veinte años. El MSX está en un momento de forma envidiable, salen títulos con una frecuencia cada vez mayor y cada vez con más calidad.



Jugabilidad:	7
Música:	8
SFX:	6
GFX:	8
Duración:	8
Extras:	6
Total:	**7**

Phantomas' Saga Infinity nos quedará en el recuerdo como una conversión ejemplar para MSX. Un juego decente de plataformas que pese a estar lejos de joyas como el Auf Wiedersehen Monty no nos hará ascos. Merece la pena echar unas partiditas hasta cogerle el truco.

Desarrollo cruzado en PC para MSX

Por David Lucena

Son diversos los factores que afectan a la hora de tomar una decisión cada vez que un programador desea comenzar un proyecto. Por supuesto, la plataforma para la que se va a realizar el mismo es uno de los más importantes, pero también influye el tipo de proyecto, el lenguaje de programación deseado, disponible o bien al que se tiene mayor afinidad, así como, por qué no, la plataforma de desarrollo; y es que no todos los dispositivos programables, ya sean computadores u otros, disponen de los requisitos necesarios para realizar el desarrollo usando el mismo como plataforma. Un ejemplo de esto son los dispositivos móviles.

A la hora de realizar un proyecto para MSX, plataforma que tratamos en esta revista, estos factores también hacen acto de presencia. A la hora de decidir si programar directamente desde ella o bien desde otra plataforma, entra en juego el criterio de cada uno. Desde mi punto de vista, tanto la comodidad como los recursos son muy importantes para el programador. La velocidad de desarrollo y de generación de elementos intermedios hasta obtener el objeto final es también factor importante tanto para el programador como para la empresa, pues a más rapidez mayor beneficio, o mejor dicho, menos gasto en inversión de tiempo que se traduce en dinero, y en el caso de un particular sin ánimo de lucro, en un mejor aprovechamiento del ocio personal. Puesto que probablemente ninguna empresa va a pagar por nuestro software para MSX, nos queda optar por la comodidad. Hay quien considera más cómodo o placentero usar la máquina original, les proporciona un sentimiento romántico de vivir en una época pasada, sintiendo que aún la están viviendo. No importa la velocidad de generación, ni la rapidez de ejecución o comodidad de depuración, sino la sensación. Por mi parte, aun teniendo parte de esa sensación prefiero la versatilidad que ofrece una plataforma superior, desde la que se puede generar el programa final con mayor rapidez, con gran comodidad al tener diversos archivos de código fuente abierto, más seguridad a la hora de poder usar repositorios o dispositivos de almacenamiento para realizar backups periódicos, efectivos y veloces. Es por ello que en este artículo trataré de exponer algunas herramientas para usar en PC, como plataforma para el desarrollo de aplicaciones para nuestro MSX.

El lenguaje de programación usado será C, si bien contendrá en muchas ocasiones apoyo de código ensamblador embebido en el propio código fuente de C. El compilador de C usado será el SDCC, elegido por estar en avanzado desarrollo y ser de código abierto, por lo que podemos disfrutar de él de forma gratuita y en diversas plataformas. El entorno operativo usado es MinGW con MSys bajo un sistema operativo Windows. El entorno MSys ha sido elegido para compatibilizar con versiones de sistema operativo Linux, pero no puedo garantizar la compatibilidad puesto que no lo he probado personalmente. Para la generación completa del programa, además, hago uso del sistema de compilación make de GNU, puesto que facilita la generación del programa, su posterior ejecución, la organización de pasos intermedios y la limpieza de archivos generados no útiles. En contra de lo que alguien pudiera pensar, adelanto que previo paso de compilación realizo un paso de preprocesado del código. Quizás alguien pueda pensar que esto no tiene sentido, pues SDCC ya preprocesa el código C, pero yo lo considero importante puesto que en proyectos grandes podemos equivocarnos en alguna definición de preproceso y no saber exactamente porqué nuestro código final hace lo que no debería hacer. Al generar el código ya preprocesado en otro directorio, podemos examinarlo antes de realizar una depuración del código ejecutable para averiguar el error. Por otra parte decir que no he usado el preprocesador de C de GNU (cpp) para generar el código preprocesado, sino el procesador de macros M4, también de GNU; aunque a primera vista pueda resultar complicado, la verdad es que ofrece elementos interesantes. Todo lo anteriormente expuesto será explicado, por lo que no es necesario alarmarse.

Lo primero es instalar MSYS, que ya contiene el conjunto de librerías MinGW, aunque quizá no las últimas versiones. La versión MSYS con la que se ha realizado el desarrollo ha sido la versión 1.0.10, descargable desde el enlace *http://prdownloads.sf.net/mingw/MSYS-1.0.10.exe?download* o bien navegando desde la página *http://www.mingw.org*

Una vez descargado e instalado, tendremos el siguiente icono en nuestra pantalla:



Al hacer doble clic aparecerá una consola similar a la de un sistema operativo Linux, tanto en forma como en comportamiento, puesto que es precisamente lo que este programa pretende simular. Así podremos ejecutar comandos como ls para listar los ficheros, cd para cambiar de directorio, etc.

Seguidamente podemos proceder a instalar el compilador cruzado de C SDCC. Para ello hay que descargarlo desde *http://prdownloads.sourceforge.net/sdcc/sdcc-2.5.0-i586-mingw32msvc.zip?download* aunque probablemente la versión actual, la 2.6 también funcione. Siempre podréis buscar las versiones desde *http://sdcc.sourceforge.net/*

Instalado también el compilador sólo queda ponerse al trabajo, pues el procesador de macros m4 viene incluido en el MSYS. Es conveniente introducir la estructura de directorios básica de nuestro proyecto, que puede ser extensible y modificada si se quiere, pero que es común en muchos casos.

Existe un directorio raíz de donde cuelgan el resto de directorios y un fichero Makefile. Básicamente no tienen que haber ficheros aparte de Makefile en el directorio raíz, si bien algún leeme.txt o algún otro archivo de copyright o ayuda inicial que no tenga que ver realmente con la generación de ficheros de proyecto. Dentro de este directorio raíz, el siguiente directorio que cobra importancia es src, pues es el directorio donde se encuentra nuestro código fuente. Después está pre, donde irá a parar el código de src que ha sido preprocesado. Un directorio intermedio dep nos indicará para cada fichero de src cuáles de éstos deben ser preprocesados y dejados en pre. Después de preprocesar estos archivos, habrá que compilarlos y el codigo objeto intermedio se depositará en obj. Finalmente, el archivo generado, el que realmente interesa será copiado a bin. Otros directorios que existen es el tools donde se encuentran las herramientas que serán usadas en procesos intermedios para alcanzar el objetivo final, o bien para hacerle un tratamiento posterior. Por último hay un directorio llamado startup que contiene el código de inicialización de nuestro programa dependiendo de si éste va a ser generado en modo rom, o para msx-dos. Además para este último existen dos variantes, los programas que van a hacer uso de los argumentos de entrada y los que no, ya que si un programa no necesita parámetros de entrada merece la pena ahorrar un poco y no introducir el código necesario para tratar estos parámetros durante la ejecución.

A continuación explicaremos cómo crear un fichero Makefile para que el comando make pueda generar un simple programa que podremos incluir en un disquete y ejecutarlo bajo MSX-DOS. Podréis acceder a la ayuda de make en *http://www.gnu.org/software/make/manual/make.html*

Un archivo Makefile consiste básicamente en un conjunto de reglas. Dichas reglas se ejecutan en un orden concreto para obtener un resultado final. Cada regla se compone de un objetivo y unos prerrequisitos además de un conjunto de comandos. Los prerrequisitos para el objetivo pueden estar vacíos así como los comandos. Cada vez que make debe procesar una regla, comprueba si el objetivo existe y si así es si está actualizado, es decir, si su creación es posterior a la de los prerrequisitos. Si no ha sido generado o no está actualizado, procede a evaluar los prerrequisitos. Cada prerrequisito puede a su vez ser generado por alguna otra regla. Cuando se han generado los prerrequisitos o se han actualizado, se ejecutan los comandos de la regla. Estos comandos son ejecutados por el intérprete de comandos, y en el caso de MSys es el intérprete bash. Normalmente los comandos son usados para generar o actualizar los objetivos. En el caso de que no existan reglas para actualizar un prerrequisito, make proporcionará un error diciendo que no puede encontrar la regla para actualizar un prerrequisito necesitado por la regla actual. Si una regla que se ejecuta tiene objetivos pero no prerrequisitos, se considerará que los prerrequisitos ya están actualizados y se procederá a ejecutar los comandos. Si una regla no tiene comandos, cuando se evalúen los prerrequisitos se considera la regla como ejecutada.

Los objetivos de estas reglas son interpretados por defecto por make como ficheros. Así pues, si make detecta que el fichero especificado en el objetivo no está creado, analiza los prerrequisitos para evaluarlos y seguidamente ejecuta los comandos. Si el fichero sí existe, comprueba que su fecha de creación sea posterior a la de los prerrequisitos, lo que demuestra que ya están actualizados. Los prerrequisitos suelen también ser ficheros, aunque como veremos más adelante pueden también no serlos.

Supongamos que queremos generar un fichero objeto. Dicho fichero dependerá de un fichero fuente, y quizá incluso de unos ficheros de cabecera.

```
programa.o: programa.c programa.h
	gcc -o programa.o -c programa.c
```

El objetivo de esta regla es programa.o. Los prerrequisitos son programa.c y programa.h. Los prerrequisitos se separan del objetivo mediante el símbolo ':'. El comando de generación del objetivo es gcc -o programa.o -c programa.c. Para distinguir una regla de un comando, éstos últimos deben estar tabulados usando la tecla de tabulación, y no espacios.

Supongamos que tenemos un fichero programa.c y otro programa.h. Cuando invocamos make en un directorio donde existe un fichero Makefile con el contenido de arriba, make ejecuta la regla. Comprueba que existe programa.o. Al no ser así, comprueba que existan programa.c y programa.h. Puesto que eso es cierto entonces procede a ejecutar el comando. Este comando invoca al compilador gcc, diciendo que la salida de la compilación (parámetro -o) sea almacenada en

programa.o, mediante la compilación (parámetro -c) del fichero programa.c.

Si volvemos a invocar a make, puesto que programa.c y programa.h no han sido modificados, se comprobará como programa.o ya está actualizado y no se ejecutarán los comandos de la regla. Si modificamos programa.c o programa.h y guardamos los cambios, al invocar make de nuevo, se volverá a ejecutar la regla y los comandos, ya que algunos archivos que forman los prerrequisitos es más nuevo que el objetivo.

Una vez introducidas las reglas, explicaremos el uso de las variables. Existen principalmente dos formas de asignar valores a una variable, la inmediata (variable := valor) y la diferida (variable = valor). En el primer modo la variable toma el valor en el momento de la asignación. En el segundo modo la variable toma el valor únicamente cuando va a ser utilizada, es decir, cuando se va a obtener el valor de la misma para un uso. Para obtener el valor de una variable, debemos de introducir el nombre de esta entre los símbolos $(). Ejemplos:

Ejemplo 1(contenido de fichero Makefile):

```
VARIABLE_A := 1
VARIABLE_B := $(VARIABLE_A)
VARIABLE_C = $(VARIABLE_A)
VARIABLE_A := 3

test:
	@echo $(VARIABLE_A)
	@echo $(VARIABLE_B)
	@echo $(VARIABLE_C)
```

Resultado 1 tras invocar make (make test):

1
2
3

En este caso el comportamiento de la variable asignada de forma diferida no es distinto del de las demás.

Ejemplo 2 (contenido de fichero Makefile):

```
VARIABLE_A := 1
VARIABLE_B := $(VARIABLE_A)
VARIABLE_C = $(VARIABLE_A)
VARIABLE_A := 3

test:
	@echo $(VARIABLE_A)
	@echo $(VARIABLE_B)
	@echo $(VARIABLE_C)
```

Resultado 2 tras invocar make (make test):

3
1
3

Como nota adicional, el símbolo '@' se usa para evitar que se escriba el comando por la salida. Habéis podido comprobar el funcionamiento de ambos tipos de asignación y los resultados obtenidos. También se ha podido ver cómo la regla con objetivo test, al no tener prerrequisitos, ejecuta automáticamente los comandos.

Es momento de empezar a crear nuestro fichero Makefile, y para ello tenemos que definir una serie de variables que se irán usando para definir otras variables y para ser usadas en algunas reglas.

```
ifeq (0, $(MAKELEVEL))
ifeq ($(OSTYPE), msys)
DRIVE          := c:
else
DRIVE          := /c
endif
ROOT_DIR       := $(shell pwd)
ROOT_DIR       := $(subst /c,$(DRIVE),$(ROOT_DIR))
BIN_INFIX      := bin
SRC_INFIX      := src
PRE_INFIX      := pre
OBJ_INFIX      := obj
DEP_INFIX      := dep
TOOLS_INFIX    := tools
STARTUP_INFIX  := startup
BIN_DIR        := $(ROOT_DIR)/$(BIN_INFIX)
SRC_DIR        := $(ROOT_DIR)/$(SRC_INFIX)
PRE_DIR        := $(ROOT_DIR)/$(PRE_INFIX)
OBJ_DIR        := $(ROOT_DIR)/$(OBJ_INFIX)
DEP_DIR        := $(ROOT_DIR)/$(DEP_INFIX)
TOOLS_DIR      := $(ROOT_DIR)/$(TOOLS_INFIX)
STARTUP_DIR    := $(ROOT_DIR)/$(STARTUP_INFIX)
export STARTUP_DIR
LIB_DIR        := $(ROOT_DIR)/lib
EMU_IN_DIR     := $(ROOT_DIR)/toemu
endif
```

Además de las definiciones de variables podemos observar algunas palabras claves interesantes: ifeq, else y endif.

Muchos ya habrán deducido el funcionamiento de estas palabras. Si queremos definir unas variables, o incluso unas reglas dependiendo de ciertas condiciones, en make se realiza mediante estas palabras clave. Si las variables o valores encerrados entre paréntesis y separados por una coma en ifeq

son iguales, se procede a evaluar el contenido hasta encontrar un endif. Si no se cumple la condición, se ignora lo que hay entre ifeq y endif.

Si usamos el trío ifeq, else y endif cuando la comparación sea cierta se ejecutará lo que hay entre ifeq y else, y en caso contrario lo que hay entre else y endif. También existe la negación de ifeq, que es ifneq, y que consiste en que la comparación de variables sea distinta para que se cumpla la condición.

En el código de arriba, definimos ese conjunto de variables sólo si la variable MAKELEVEL es cero, es decir, si el nivel de ejecución de make es el inicial. Cuando se invoca a make por primera vez, la variable MAKELEVEL es cero, pero si dentro del fichero Makefile existe alguna regla que invoca a make de nuevo, entonces esta variable se incrementa. No queremos que se redefinan de nuevo ciertas variables por ser susceptible de causar errores.

Hay otra variable que se define dependiendo del valor de OSTYPE. Esta variable toma el valor msys en el caso de estar ejecutándose make bajo este entorno, ya que OSTYPE se define como una variable de entorno. Así pues, supongo que en este caso se ejecuta bajo Windows, y conviene definir la unidad donde se encuentra el proyecto en formato Windows para evitar problemas con algunas aplicaciones. En el caso de que no se haya definido OSTYPE, se supone que el entorno es Linux. Para usuarios de Mac no he tenido en cuenta ninguna alternativa, así que tendrán que buscarse las alubias.

Más cosas interesantes que podemos observar radican en la variable ROOT_DIR. Esta variable toma un valor que proviene del resultado de la ejecución de un comando del intérprete de comandos. Mediante $(shell comando), podemos obtener el valor generado por el comando comando. En este caso, pwd devuelve el directorio actual, desde el que se está ejecutando make. Más adelante esta variable toma otro valor: $(subst /c,$(DRIVE),$(ROOT_DIR)). Lo que realiza esta función de make es sustituir una ocurrencia de una cadena por otra en una variable. En este caso, se sustituye /c por el contenido de la variable DRIVE en la variable ROOT_DIR. Así conseguimos formar un directorio raíz de proyecto correcto.

Vemos también que existe la línea export STARTUP. Con export podemos exportar la variable y su valor al intérprete de comandos, para que subsecuentes llamadas a make o a otras aplicaciones y comandos del intérprete de comandos (shell) puedan ver esa variable y su valor.

Sin embargo hasta aquí no ha llegado la definición de variables. Existen más. Algunas de ellas no se usan en este momento pero en un futuro podrían ser útiles y tampoco molestan estando ahí.

```
ROM             := 0
BASICROM        := 1
MSXDOS          := 2
MSXDOSARG       := 3
# MSXDOS, MSXDOSARG, ROM, BASICROM
OUTPUT_TYPE  := ROM

src_files = $(wildcard $(SRC)/*.c)

PLATFORM_SHORT_NAME             := MSX
# WINNT, LINUX
CROSSPLATFORM_SHORT_NAME        := WINNT
COMPILER_EXECUTABLE             := sdcc.exe
COMPILER_EXECUTABLE_PATH        := $(DRIVE)/prog/compilers/sdcc/bin
COMPILER_BASE_PATH              := $(DRIVE)/prog/compilers/sdcc
ASSEMBLER_EXECUTABLE            := as-z80.exe
ASSEMBLER_EXECUTABLE_PATH       := $(DRIVE)/prog/compilers/sdcc/bin
ASSEMBLER_BASE_PATH             := $(DRIVE)/prog/compilers/sdcc
LINKER_EXECUTABLE               := link-z80.exe
LINKER_EXECUTABLE_PATH          := $(DRIVE)/prog/compilers/sdcc/bin
LINKER_BASE_PATH                := $(DRIVE)/prog/compilers/sdcc
# OPENMSX, BLUEMSX
EMULATOR_SHORT_NAME             := OPENMSX
ifeq (BLUEMSX, $(EMULATOR_SHORT_NAME))
 EMULATOR_EXECUTABLE            := blueMSX.exe
 EMULATOR_EXECUTABLE_PATH       := $(DRIVE)/proxp/emula/bluemsx
 EMULATOR_BASE_PATH             := $(DRIVE)/proxp/emula/bluemsx
 MACHINE_NAME                   :=
 MACHINE_PARAM                  :=
 ROMA_PARAM                     := /rom1
 ROMB_PARAM                     := /rom2
 DISKA_PARAM                    := /diskA
 DISKB_PARAM                    := /diskB
```

```
endif
ifeq (OPENMSX, $(EMULATOR_SHORT_NAME))
 EMULATOR_EXECUTABLE            := openmsx.exe
 EMULATOR_EXECUTABLE_PATH       := $(DRIVE)/proxp/emula/openmsx
 EMULATOR_BASE_PATH             := $(DRIVE)/proxp/emula/openmsx
 MACHINE_NAME                   := Philips_NMS_8255
 MACHINE_PARAM                  := -machine
 ROMA_PARAM                     := -carta
 ROMB_PARAM                     := -cartb
 DISKA_PARAM                    := -diska
 DISKB_PARAM                    := -diskb
endif
DSK_FILE_NAME                   := example.dsk
DSK_FILES                       :=
USE_OWN_LINKER                  := true
ifeq ($(OUTPUT_TYPE), MSXDOS)
 STARTUP_CODE_BASEFILENAME      := crt0msx_msxdos
 STARTUP_CODE_FILENAME          := crt0msx_msxdos.o
 DATA_LOC                       := 0x0000
 CODE_LOC                       := 0x0106
 CFLAGS                         := -mz80 --code-loc $(CODE_LOC) --data-loc $(DATA_LOC) --no-std-crt0
 TARGET_ROM_OR_DSK              :=
 EXE_EXT                        := com
endif
ifeq ($(OUTPUT_TYPE), MSXDOSARG)
 STARTUP_CODE_BASEFILENAME      := crt0msx_msxdosarg
 STARTUP_CODE_FILENAME          := crt0msx_arg.o
 DATA_LOC                       := 0x0000
 CODE_LOC                       := 0x0170
 CFLAGS                         := -mz80 --code-loc $(CODE_LOC) --data-loc $(DATA_LOC) --no-std-crt0
 TARGET_ROM_OR_DSK              :=
 EXE_EXT                        := com
endif
ifeq ($(OUTPUT_TYPE), ROM)
 STARTUP_CODE_FILENAME          := crt0msx_rom.o
 DATA_LOC                       := 0x0000
 CODE_LOC                       := 0x4020
 CFLAGS                         := -mz80 --code-loc $(CODE_LOC) --data-loc $(DATA_LOC) --no-std-crt0
 TARGET_ROM_OR_DSK              :=
 EXE_EXT                        := rom
endif
LDFLAGS                         := -n
# Custom targets -->
TARGET_PREBUILDFILES            := prebuildfiles$(PLATFORM_SHORT_NAME)
TARGET_BUILDFILES               := buildfiles$(PLATFORM_SHORT_NAME)
TARGET_POSTBUILDFILES           := postbuildfiles$(PLATFORM_SHORT_NAME)
TARGET_RUNFILES                 := runfiles$(PLATFORM_SHORT_NAME)
TARGET_CLEANPREBUILTFILES       := cleanprebuiltfiles$(PLATFORM_SHORT_NAME)
# Custom targets <--

APP_NAME                        := testmake
APP_EXE                         := $(APP_NAME).$(EXE_EXT)
BUILD_TARGET                    := $(APP_EXE)

# Actual Makefile variables -->
# Actual Makefile variables <--
CC := $(COMPILER_EXECUTABLE_PATH)/$(COMPILER_EXECUTABLE)
export CC
AS := $(ASSEMBLER_EXECUTABLE_PATH)/$(ASSEMBLER_EXECUTABLE)
export AS
ifeq (true,$(USE_OWN_LINKER))
LD := $(LINKER_EXECUTABLE_PATH)/$(LINKER_EXECUTABLE)
else
LD := $(CC)
endif
CPP := cpp

# Files
```

```
H_FILES_FULL           := $(wildcard $(SRC_DIR)/*.h)
H_FILES                := $(subst $(SRC_DIR)/,,$(H_FILES_FULL))
HPP_FILES_FULL         := $(wildcard $(SRC_DIR)/*.hpp)
HPP_FILES              := $(subst $(SRC_DIR)/,,$(HPP_FILES_FULL))
C_FILES_FULL           := $(wildcard $(SRC_DIR)/*.c)
C_FILES_FULL_NO_EXT    := $(C_FILES_FULL:%.c=%)
C_FILES                := $(subst $(SRC_DIR)/,,$(C_FILES_FULL))
C_FILES_NO_EXT         := $(subst $(SRC_DIR)/,,$(C_FILES_FULL_NO_EXT))
CPP_FILES_FULL         := $(wildcard $(SRC_DIR)/*.cpp)
CPP_FILES_FULL_NO_EXT  := $(CPP_FILES_FULL:%.cpp=%)
CPP_FILES              := $(subst $(SRC_DIR)/,,$(CPP_FILES_FULL))
CPP_FILES_NO_EXT       := $(subst $(SRC_DIR)/,,$(CPP_FILES_FULL_NO_EXT))
ASM_FILES_FULL         := $(wildcard $(SRC_DIR)/*.asm)
ASM_FILES_FULL_NO_EXT  := $(ASM_FILES_FULL:%.asm=%)
ASM_FILES              := $(subst $(SRC_DIR)/,,$(ASM_FILES_FULL))
ASM_FILES_NO_EXT       := $(subst $(SRC_DIR)/,,$(ASM_FILES_FULL_NO_EXT))
SRC_FILES_FULL         := $(C_FILES_FULL) $(CPP_FILES_FULL) $(ASM_FILES_FULL)
INC_FILES_FULL         := $(H_FILES_FULL) $(HPP_FILES_FULL)
SRC_FILES_FULL         := $(C_FILES_FULL) $(CPP_FILES_FULL) $(ASM_FILES_FULL)
SRC_FILES_FULL_NO_EXT  := $(C_FILES_FULL_NO_EXT) $(CPP_FILES_FULL_NO_EXT) $(ASM_FILES_FULL_NO_EXT)
MAKFILE_VARS_FILE_FULL := $(DEP_DIR)/vars.d

PRE_FILES_FULL         := $(subst $(SRC_INFIX),$(PRE_INFIX),$(SRC_FILES_FULL))
DEP_FILES_FULL         := $(subst $(SRC_INFIX),$(DEP_INFIX),$(SRC_FILES_FULL:%.c=%.P))
OBJ_FILES_FULL         := $(subst $(SRC_INFIX),$(OBJ_INFIX),$(SRC_FILES_FULL:%.c=%.o))
OBJ_FILES_FULL_NO_EXT  := $(subst $(SRC_INFIX),$(OBJ_INFIX),$(SRC_FILES_FULL_NO_EXT:%.c=%.o))

df = $(DEP_DIR)/$(*F)
of = $(OBJ_DIR)/$(*F)

DEFINEDVARS            := ROM BASICROM MSXDOS MSXDOSARG OUTPUT_TYPE
EXPANDEDDEFINEDVARS    := $(foreach var, $(DEFINEDVARS),$(var)=$($(var)))

M4DEFINES              := $(foreach varandvalue, $(EXPANDEDDEFINEDVARS),--define=$(varandvalue))
M4FLAGS                := --prefix-builtins --debug=
M4COMMANDLINE  = BASHM4DEFS=`head --lines=1 $(MAKFILE_VARS_FILE_FULL)`;m4 $$BASHM4DEFS $(M4FLAGS)
--include=$(SRC_DIR) $< >$@

# Note the last parameter doest no work properly because BASHM4DEFS is assigned directly instead
of written without modification
MAKEDEPEND = awk -f $(TOOLS_DIR)/makepredepend/makepredepend.awk $(SRC_DIR) $(PRE_DIR) $(DEP_DIR)
$< "$(M4COMMANDLINE);"
BUILDMAKEFILEVARSFILE = \
        @if ! test -e $(MAKFILE_VARS_FILE_FULL); then
                        \
                echo Building $(MAKFILE_VARS_FILE_FULL);
                        \
                echo "$(M4DEFINES)" > $(MAKFILE_VARS_FILE_FULL);
                \
        else
                                                  \
                if [ "$(M4DEFINES)" != "$$(head --lines=1 $(MAKFILE_VARS_FILE_FULL))" ]; then     \
                        echo Updating $(MAKFILE_VARS_FILE_FULL);
                        \
                        echo "$(M4DEFINES)" > $(MAKFILE_VARS_FILE_FULL);
                \
                fi;
                                                          \
        fi;
```

Hay muchas variables que explicar, pero lo haremos a medida que se vayan creando las reglas. Primero veremos la regla por defecto.

```
# Default rule
all: build run
```

Si queremos que make use una regla por defecto como inicial, tenemos que definir all, y como requisitos aquellos objetivos de las reglas que queremos que se ejecuten. En nuestro caso, por defecto se creará el ejecutable del proyecto y seguidamente se invocará un emulador.

```
build: build_prepare buildtools preprocess prebuildfiles buildfiles $(TARGET_POSTBUILDFILES)

run: $(TARGET_RUNFILES)
```

Algunas de las reglas contienen prerrequisitos que dependen de la plataforma para la que se vaya a crear el proyecto, aunque en nuestro caso por ahora es MSX, como se puede ver en la variable PLATFORM_SHORT_NAME.

Algunas reglas no las imprimiremos, aunque daremos una breve explicación. Por ejemplo build_prepare crea los directorios necesarios para generar las salidas y los archivos intermedios. En el caso de buildtools se generan los ejecutables de las herramientas que usaremos más adelante para convertir archivos intermedios a un formato entendible por la máquina, y otros procesos posteriores necesarios.

Un paso interesante es el que tiene que ver con preprocess. La regla completa es la siguiente:

```
preprocess: preprocess_prepare buildmakefilevarsfile $(PRE_FILES_FULL)
```

Con esta regla se pretende preprocesar los archivos de código fuente y dejarlos ya preprocesados en un directorio intermedio, concretamente pre dentro del directorio raíz del proyecto. Para ello primero se genera un fichero con unas variables determinadas del fichero Makefile actual y el valor en que tienen en el momento de ejecutar make. Esto es así para que a la hora de preprocesar los ficheros, se tengan en cuenta estos cambios. Como prerrequisito de preprocess observamos que también existe la lista de ficheros que tiene que ser generada, y esa lista de ficheros se hace referencia en la siguiente regla:

```
$(PRE_DIR)/%.c: $(SRC_DIR)/%.c $(MAKFILE_VARS_FILE_FULL)
        @echo Building Generic $<
        @$(MAKEDEPEND)
        $(M4COMMANDLINE)
```

El comodín %.c se refiere a todo fichero de extensión .c. Así pues, esta regla se activa cada vez que se encuentra que una regla, en este caso la regla preprocess hace referencia en sus prerrequisitos a un fichero que coincide con el especificado por el comodín. Cada fichero ya preprocesado requiere el código fuente y además el fichero que mencionábamos de variables de Makefile, como por ejemplo el modo de compilación de proyecto, es decir, si queremos generar una ROM, o un archivo .COM, etc.

Una vez se comprueba la existencia de esos prerrequisitos, se procede a la generación del fichero. Primeramente se genera un fichero de dependencias para el fichero de salida. Cada fichero fuente tiene un número determinado de archivos de los que depende, pues son incluidos dentro de él en el momento de ser preprocesados. Lo que hace MAKEDEPEND es generar estas dependencias en un fichero, que seguidamente se incluirá en el propio Makefile mediante la palabra clave -include. De esta manera cada vez que se modifique un fichero que se haya incluido en el archivo que se tiene intención de preprocesar, sabremos que el archivo a preprocesar debe ser tratado a la fuerza, porque un cambio en los ficheros que incluye representa un cambio en él. Esto se hace así porque no todos los ficheros cambian en cada compilación. Cada vez que realizamos un cambio, añadimos código o corregimos un error, no todos los ficheros cambian, y preprocesar y compilar cada uno de ellos es un tiempo de proceso valioso. Cuando el proyecto crece, esto se nota, por lo que se hace necesario poner reglas para realizar el trabajo de generación mínimo posible y que así obtengamos el resultado lo más rápido que se pueda.

Ahora sólo falta el preprocesado en sí. Para ello llamamos a m4 mediante M4COMMANDLINE, que genera el comando de preprocesado.

```
-include $(DEP_FILES_FULL)
```

Con esto acabamos el preprocesado, incluyendo los ficheros de dependencia para cada fichero que va a ir a parar en el directorio de ficheros preprocesados. Un ejemplo de lo que contiene un fichero de dependencias lo podemos ver a continuación para el archivo debug.c.

```
c:/prog/projects/msx/testmake/pre/debug.c: c:/prog/projects/msx/testmake/src/debug.c
c:/prog/projects/msx/testmake/src/debug.h c:/prog/projects/msx/testmake/src/mr_conio_inc.h c:/prog/proj-
ects/msx/testmake/src/system_vars_inc.h
c:/prog/projects/msx/testmake/dep/vars.d
  @echo " Automatic building for "c:/prog/projects/msx/testmake/pre/debug.c
  @BASHM4DEFS=`head --lines=1 c:/prog/projects/msx/testmake/dep/vars.d`;m4 $$BASHM4DEFS $(M4FLAGS) --
include=c:/prog/projects/msx/testmake/src $< >$@;
c:/prog/projects/msx/testmake/src/debug.h:
c:/prog/projects/msx/testmake/src/mr_conio_inc.h:
c:/prog/projects/msx/testmake/src/system_vars_inc.h:
```

La herramienta que calcula las dependencias, un script en awk, se encarga de recorrer el fichero debug.c del directorio de archivos fuente root_dir/src y detectar todos los archivos que incluye. Más tarde, al ejecutar make, si ningún archivo ha cambiado, no ocurre nada, en caso contrario preprocesa el archivo con las definiciones iniciales del fichero Makefile a las

que hicimos mención anteriormente.

Ahora debemos generar los ficheros objeto, los .obj. Esta regla es sencilla y simplemente invoca al compilador cruzado sdcc.exe diciendo que compile los ficheros, pero sin enlazarlos en un archivo final.

El archivo binario final que se creará, se hace mediante la creación de un fichero intermedio de enlazado, en el que se escriben unos parámetros necesarios para el enlazador de ficheros objeto, así como los ficheros objetos que deben ser enlazados y el nombre del fichero binario final.

```
$(BIN_DIR)/$(APP_NAME).ihx: buildobjects
        @echo -Creating link file
        @echo -- > $(BIN_DIR)/$(APP_NAME).lnk
        @echo -m >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -j >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -x >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -i $(BIN_DIR)/$(APP_NAME) >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -b _CODE = $(CODE_LOC) >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -b _DATA = $(DATA_LOC) >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo "-k c:\prog\compilers\sdcc\bin\..\lib\z80" >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -l z80 >> $(BIN_DIR)/$(APP_NAME).lnk
        @for file in $(OBJ_FILES_FULL); do echo $$file >> $(BIN_DIR)/$(APP_NAME).lnk; done;
        @echo $(STARTUP_DIR)/bin/$(STARTUP_CODE_FILENAME) >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo "-l z80.lib" >> $(BIN_DIR)/$(APP_NAME).lnk
        @echo -Linking
        @$(LD) $(LDFLAGS) -f $(BIN_DIR)/$(APP_NAME).lnk
        @echo -Done
```

El archivo binario final se debe convertir a un formato entendible para la plataforma de destino, y es lo que se hace mediante la regla cuyo objetivo es binarize. Finalmente se postprocesa en caso necesario mediante herramientas externas y finalmente se llama al emulador.

Existen otras reglas que se usan para limpiar los archivos intermedios generados, otros que simplemente hacen de intermediarios para que la salida durante la ejecución de make sea un poco esclarecedora y en el caso de que se produzca un error en el proceso sepamos con cierta precisión en qué momento ha fallado.

Creo que el artículo ya tiene un cuerpo importante, y para no cansar al lector y dejarle con la incógnita de lo que realmente interesa, el código fuente, pospondré la finalización para un próximo número. De esta manera tendré espacio para entretenerme en detalles y existirá un tiempo prudencial en el que los lectores pueden hacer preguntas sobre dudas de este artículo que podrían ser aclaradas en el siguiente.

Os dejo como podéis ver el resultado de la generación del programa, así como una pantalla de la salida del mismo. Evidentemente ahora no podemos ver nada que sea extraordinario, pero os aseguro que os he allanado el terreno para que podáis hacer grandes proyectos con vuestro msx, o también pequeños. El caso es que la estructura que os he definido os servirá de esqueleto para muchas de las cosas que podáis imaginar, y si existe algo que aún no se puede hacer, es muy probable que el sistema se pueda extender para cubrir esa falta. Por ejemplo sin ir más lejos ahora mismo todo el ensamblador que queráis introducir debéis hacerlo dentro de un archivo .c, pero es posible crear reglas para compilarlos por separado y enlazarlos en el ejecutable final. También se pueden procesar ficheros gráficos para convertirlos a pantallas de msx y luego integrarlas en un fichero con todos los datos estáticos del programa, como niveles, músicas, textos, etc. Esto es el inicio de algo que puede crecer tanto como queráis y de una forma no muy complicada una vez te has adaptado a la nomenclatura y el funcionamiento. Espero que os sirva de utilidad y, por qué no, que podamos hacer que juntos el sistema crezca en los próximos números para cubrir todas esas necesidades que se os van apareciendo.

Un enorme saludo.

David Lucena

Ejemplo: ¡Saludos al lector!



Aritmética de punto fijo

Por Avelino Herrera

La aritmética de punto fijo permite realizar cálculos con números reales utilizando unidades de cálculo entero. Frente al cálculo con coma flotante (o punto flotante) que obtiene resultados más precisos a costa de un menor rendimiento, tenemos el cálculo mediante punto fijo (o coma fija) que da resultados menos precisos pero a unas velocidades comparables al cálculo con números enteros.

Un número representado en notación de punto fijo viene determinado por un número de bits constante para la parte entera y un número de bits constante también para la parte fraccionaria. La notación matemática más utilizada para estos menesteres es la denominada notación Q. Un número QI.F denotará un número de I+F bits con I bits reservados para la parte entera y F bits reservados para la parte fraccionaria. Si para un número entero teníamos la distribución del valor de cada bit según la figura 1, para un número almacenado usando notación en punto fijo QI.F la distribución del valor de cada bit vendrá dada por la figura 2.

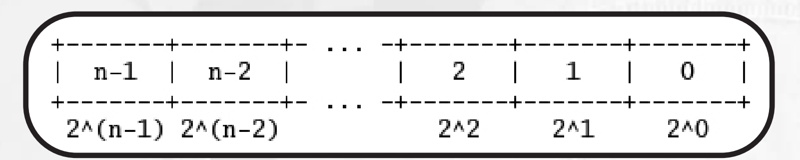


Figura 1. Un número entero de n bits

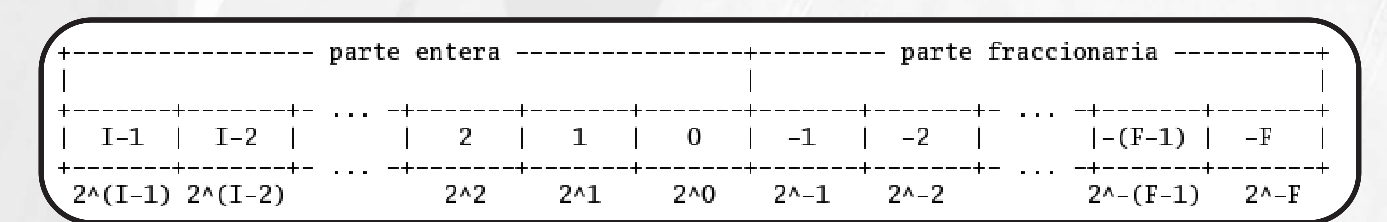


Figura 2. Un número en punto fijo QI.F

Como se puede ver un número QI.F ocupa, efectivamente I+F bits y, al contrario que la notación en coma flotante, no tenemos ningún campo de bits que nos indique dónde se halla la coma: La coma es fija. También se puede apreciar que la parte fraccionaria está ponderada siguiendo potencias negativas de 2. Imaginemos una notación Q2.6 y que queremos representar los números 1.5, 0.125 y 1.1682.

```
1.5 = 2^0 + 2^-1 = 01100000
     parte entera = 01
     parte fraccionaria = 100000
0.125 = 2^-3 = 00001000
     parte entera = 00
     parte fraccionaria = 001000
1.1682 ~= 1 + 2^-3 + 2^-5 = 1.1562
     (error de 0.01195)
     parte entera = 01
     parte fraccionaria = 001010
```

Como se puede observar cuanto mayor es el número de bits que ocupe la parte fraccionaria (F) mayor será la precisión de los números que manejemos. Se define la resolución de un número en punto fijo como:

```
e = 1 / (2^F)
```

Si despejamos la F:

F = log2(1 / e)

y como F debe ser un número natural:

F = ceil(log2(1 / e))

Luego si queremos obtener una resolución de, al menos, 0.001 deberemos usar como mínimo

F = ceil(log2(1 / 0.001)) = ceil(log2(1000))
F = ceil(9.9658) = 10 bits

OPERACIONES BÁSICAS

Los número negativos en punto fijo pueden ser representados, al igual que los enteros, mediante la notación usual de complemento a 2 y tanto la suma como la multiplicación se realizan como si fuesen números enteros. En la multiplicación hay que tener en cuenta que si tenemos dos números QA.B y QC.D y los multiplicamos entre sí, el resultado será un número en formato Q(A+C).(B+D).

SUMA:

```
1.5 + 0.125 --> 01100000 + 00001000 = 01101000
01101000 --> 2^0 + 2^-1 + 2^-3 = 1.625
```

MULTIPLICACIÓN:

```
1.5 * 0.125 --> 01100000 * 00001000 =
0000001100000000
parte entera = 0000 (2 + 2 = 4 bits)
parte fraccionaria = 001100000000 (6 + 6 = 12 bits)
resultado = 2^-3 + 2^-4 = 0.1875
```

Obsérvese que si queremos alojar el resultado de una multiplicación de nuevo en un formato QA.B ó QC.D habrá que sacrificar, tanto precisión en la parte fraccionaria, como rango en la parte entera y pueden provocarse desbordamientos.

Utilizaremos el formato Q8.8 para los ejemplos a lo largo de este artículo. Para obtener la prepresentación en formato de punto fijo de un número real podemos utilizar el siguiente programa auxiliar en C del listado 1.

```
/* LISTADO 1. Conversor de decimal a Q8.8. */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double v, w;
signed short int fixed_point;
signed char integer_part;
signed char fractional_part;

int main(int argc, char **argv) {
   v = atof(argv[1]);
   fixed_point = (signed short int)
                 round(v * 256);
   w = (double) fixed_point / 256.0;
   integer_part = 0x00FF & (fixed_point >> 8);
   fractional_part = 0x00FF & fixed_point;
   printf("original value: %lf\n", v);
   printf("fixed_point value: %d (%02X)\n",
          fixed_point, fixed_point);
   printf("integer part: %d (%02X)\n",
          integer_part, integer_part);
   printf("fractional part: %d (%02X)\n",
          fractional_part, fractional_part);
   printf("\nstored value: %lf (error = %lf)\n",
          w, fabs(w - v));
   return 0;
}
```

Invocando al programa de la forma "./aux 3.561" nos dará la representación en formato Q8.8 del número 3.561 o, al menos, su representación más aproximada.

Salvo este pequeño trozo de código auxiliar en C, utilizaré ensamblador para implementar el resto de cosas que necesitamos para manejar nuestros números Q8.8. Q8.8 implica el uso de 1 byte para la parte fraccionaria (el de la dirección de memoria más baja) y 1 byte para parte entera (el de la dirección de memoria más alta). La suma, es muy sencilla, ya que lo único que tenemos que hacer es una suma de dos números de 16 bits (ver listado 2).

```
; LISTADO 2. Suma en punto fijo.
ld hl,(sumando_1)
ld bc,(sumando_2)
add hl,bc
ld (resultado),hl
```

Para la multiplicación el concepto es el mismo, aunque el código es más extenso ya que necesitamos hacer una multiplicación entre dos números de 16 bits cada uno que dará como resultado un número de 32 bits. Además, tenemos que controlar el signo de los factores y tener siempre presente que el resultado será un número de 32 bits en formato Q16.16 (ver listado 3).

```
; LISTADO 3. Multiplicación en punto fijo.
;--------------------------------------------
; multiplicación: BCHL = BC * DE
; http://map.tni.nl/articles/mult_div_shifts.php
;
fast_mul_16x16_32:
   ld a,c
   ld c,b
   ld hl,0
   ld b,16
fast_mul_16x16_32_loop:
   add hl,hl
   rla
   rl c
   jr nc,fast_mul_16x16_32_no_add
   add hl,de
   adc a,0
   jp nc,fast_mul_16x16_32_no_add
   inc c
fast_mul_16x16_32_no_add:
   djnz fast_mul_16x16_32_loop
   ld b,c
   ld c,a
   ret
```

```
;-----------------------------------
; multiplicación 16x16 bits = 32 bits
mul_16x16_32:
   ld a,0
   ld (mul_16x16_32_negate_result),a
   ; determinamos si hay que negar el
   ; resultado y/o los multiplicandos
   ld hl,mul_16x16_32_factor1+1
   bit 7,(hl)
   jp z,mul_16x16_32_test_factor2_sign
   ; negamos el factor1
   ld hl,mul_16x16_32_factor1
   call neg_16
   ld a,(mul_16x16_32_negate_result)
   cpl
   ld (mul_16x16_32_negate_result),a
mul_16x16_32_test_factor2_sign:
   ld hl,mul_16x16_32_factor2+1
   bit 7,(hl)
   jp z,mul_16x16_32_do_mul
   ; negamos el factor2
   ld hl,mul_16x16_32_factor2
   call neg_16
   ld a,(mul_16x16_32_negate_result)
   cpl
   ld (mul_16x16_32_negate_result),a
mul_16x16_32_do_mul:
   ld bc,(mul_16x16_32_factor1)
   ld de,(mul_16x16_32_factor2)
   call fast_mul_16x16_32
   ld (mul_16x16_32_result),hl
   ld (mul_16x16_32_result+2),bc
   ; miramos a ver si hay que negar
   ; el resultado
   ld a,(mul_16x16_32_negate_result)
   or a
   jp z,mul_16x16_32_ret
   ; negamos los 16 bits de en medio del
   ; resultado, que son los que vamos a
   ; utilizar
   ld hl,mul_16x16_32_result
   call neg_32
mul_16x16_32_ret:
   ret
mul_16x16_32_negate_result:
   db 0
mul_16x16_32_factor1:
   dw 0
mul_16x16_32_factor2:
   dw 0
mul_16x16_32_result:
   ds 4,0
```

```
;----------------------
; (hl[16]) := -(hl[16])
neg_16:
   ; load fractional part
   ld a,(hl)
   or a
   jp z,neg_16_negate_integer_part
   ; negate fractional part
   neg
   ld (hl),a
   ; and do a "not" on integer part
   inc hl
   ld a,(hl)
   cpl
   ld (hl),a
   ret
neg_16_negate_integer_part:
   inc hl
   ld a,(hl)
   neg
   ld (hl),a
   ret
```

```
;----------------------
; (hl[32]) := -(hl[32])
neg_32:
   ld b,4
neg_32_prev_loop:
   ld a,(hl)
   or a
   jp nz,neg_32_prev_loop_end
   inc hl
   djnz neg_32_prev_loop
neg_32_prev_loop_end:
   ld a,b
   or a
   jp z,neg_32_ret
   ld a,(hl)
   neg
   ld (hl),a
   inc hl
neg_32_cpl_loop:
   ld a,(hl)
   cpl
   ld (hl),a
   inc hl
   djnz neg_32_cpl_loop
neg_32_ret:
   ret
```

Nótese que, si queremos obtener un Q8.8 a partir del Q16.16 que nos sale de la multiplicación, lo único que debemos hacer es coger los dos bytes centrales del número (ver listado 4). Hay que tener en cuenta que de esta forma perdemos precisión ya que al desechar el byte menos significativo estamos desechando las potencias 2^-8 a 2^-16 de la parte fraccionaria, y corremos el peligro de generar desbordamiento ya que también estamos desechando el byte más significativo de la parte entera.

```
; Listado 4. Truncar un Q16.16 a un Q8.8
ld hl,mul_16x16_32_result+1
ld de,destino_q8_8
ld b,0
ld c,2
ldir
```

En la notación Q8.8 que estamos utilizando la precisión será de e = 1 / (2^F) = 1 / (2^-8) = 0.00390625.

EJEMPLO DE USO

Vamos a utilizar nuestro código de manejo de números Q8.8 para hacer algo más que simples sumas y multiplicaciones: vamos realizar una simulación numérica de un sistema físico de segundo orden. Consideremos una masa colgante y sujeta al techo por un muelle y por un amortiguador hidráulico (ver figura 3).

"Fm" es la fuerza de oposición del muelle en Newtons, siendo "Km" la constante de proporcionalidad del muelle [Newtons/metro] e "y" la deformación [metros]. "Fa" es la fuerza de oposición del amortiguador, siendo "Ka" la constante de proporcionalidad del amortiguador [Newtons/metro/segundo] y "v" la velocidad de deformación [metros/segundo]. "P" es el peso del cuerpo, siendo "M" su masa [Kilogramos] y "g" la aceleración de la gravedad [9.8 metros/segundo^2].

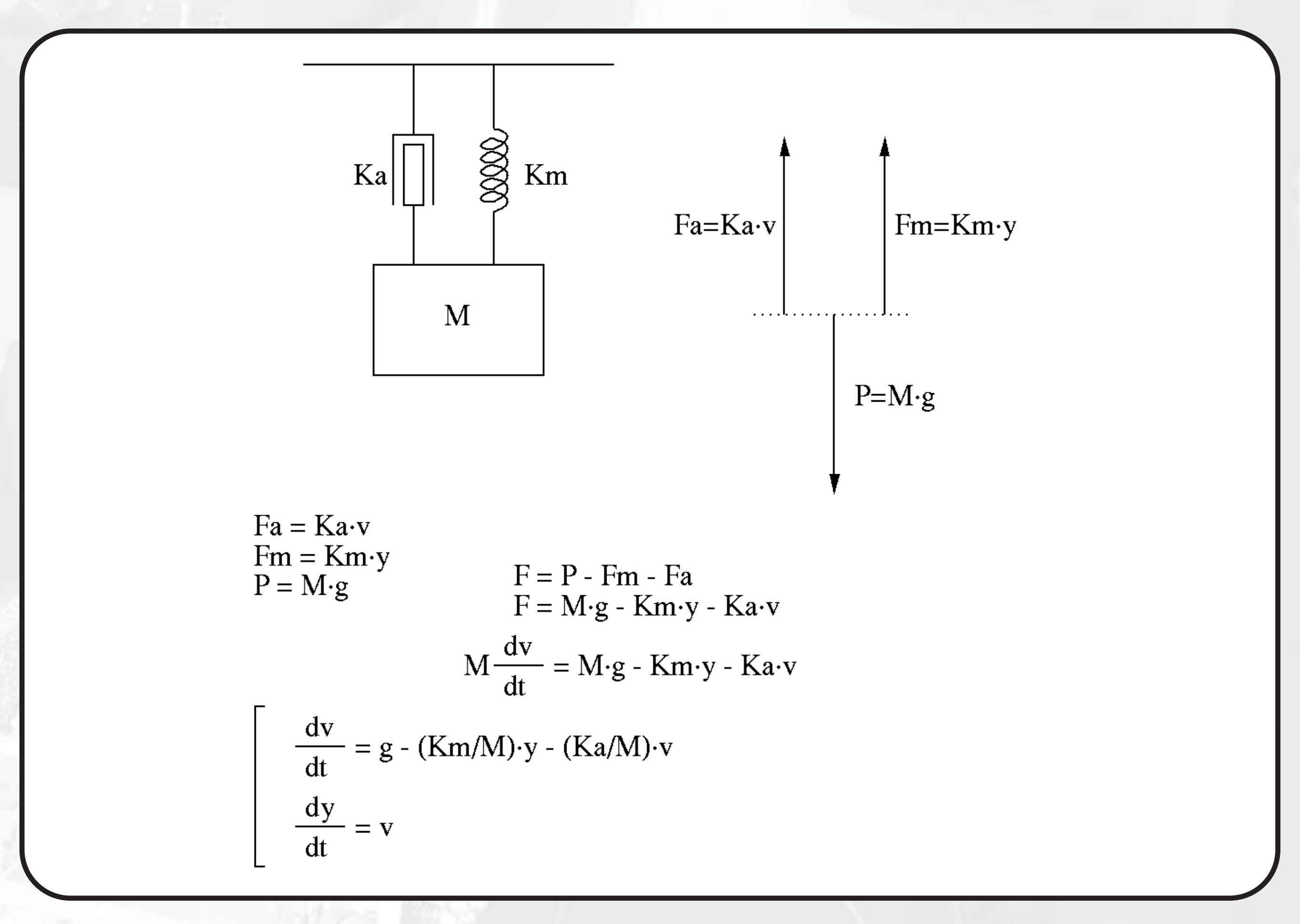


Figura 3. Sistema físico de segundo orden

Según la fórmula de Euler de aproximación de ecuaciones diferenciales (figura 4) es posible aproximar tanto como queramos la solución de una ecuación diferencial mediante diferencias finitas. Aplicándola a nuestro sistema obtenemos el siguiente sistema de dos ecuaciones en diferencias finitas:

```
v[i+1] = v[i] + H * (g - (Km/M)*y[i] - (Ka/M)*v[i])
y[i+1] = y[i] + H * v[i]
```

Cuanta más precisión queramos, más próximo a cero deberemos elegir H.

$$\frac{dy}{dt}=f(y,t)\rightarrow y_{i+1}=y_i+h\cdot f(y_i,t_i)$$

Figura 4. Fórmula de Euler

Una vez tenemos nuestro sistemas de ecuaciones diferenciales en forma de diferencia finitas ya podemos hacer la simulación de forma sencilla. En el listado 5 podemos ver la forma en que he implementado las dos ecuaciones en diferencias finitas.

```
; LISTADO 5. Rutina de simulación.
;---------------------------------------------------------------------------
; Realizamos la simulación del sistema de ecuaciones diferenciales que rige el comportamiento
; de una masa colgante enganchada a un muelle y a un amortiguador
;
;     dv                               |     dy
;    ---- = g - (Km/M)*y - (Ka/M)*v    |    ---- = v
;     dt                               |     dt
;
; Siendo "g" la aceleración de la gravedad, "Km" la constante del muelle, "Ka" la constante del
; amortiguador, "M" la masa del cuerpo, "v" la velocidad e "y" su posición. Para realizar la
; simulación utilizamos la fórmula de Euler para aproximación de ecuaciones diferenciales:
;
;    v[i+1] = v[i] + H * (g - (Km/M)*y[i] - (Ka/M)*v[i])
;    y[i+1] = y[i] + H * v[i]
;
; H es una constante de tiempo que indica lo separadas que están dos muestras consecutivas de
; v[] o de x[], cuanto más pequeña es H, mejor es la aproximación.
simulation:
   ld bc,500   ; iteramos 500 veces. 500 * H = número de segundos que abarca la simulación
   push bc
simulation_loop_start:
   ; primera ecuación diferencial
   ; v[i+1] = v[i] + H * (g - (Km/M)*y[i] - (Ka/M)*v[i])
   ; temp1 := -(Ka/M) * v_prev
   ld hl,constante_amortiguador_div_masa
   ld de,mul_16x16_32_factor1
   ld b,0
   ld c,2
   ldir
   ld hl,v_prev
   ld de,mul_16x16_32_factor2
   ld b,0
   ld c,2
   ldir
   call mul_16x16_32
   ld hl,mul_16x16_32_result+1
   ld de,temp1
   ld b,0
   ld c,2
   ldir
   ld hl,temp1
   call neg_16
   ; temp2 := -(Km/M) * y_prev
   ld hl,constante_muelle_div_masa
   ld de,mul_16x16_32_factor1
   ld b,0
   ld c,2
   ldir
   ld hl,y_prev
   ld de,mul_16x16_32_factor2
   ld b,0
   ld c,2
   ldir
   call mul_16x16_32
   ld hl,mul_16x16_32_result+1
   ld de,temp2
   ld b,0
   ld c,2
   ldir
   ld hl,temp2
   call neg_16
   ; temp3 := temp1 + temp2
   ld hl,(temp1)
   ld bc,(temp2)
   add hl,bc
   ld (temp3),hl
   ; temp3 := temp3 + g
```

```
ld hl,(temp3)
ld bc,(constante_gravedad)
add hl,bc
ld (temp3),hl
; temp3 := H * temp3
ld hl,h_euler
ld de,mul_16x16_32_factor1
ld b,0
ld c,2
ldir
ld hl,temp3
ld de,mul_16x16_32_factor2
ld b,0
ld c,2
ldir
call mul_16x16_32
ld hl,mul_16x16_32_result+1
ld de,temp3
ld b,0
ld c,2
ldir
; v := v_prev + temp3
ld hl,(v_prev)
ld bc,(temp3)
add hl,bc
ld (v),hl
; segunda ecuación diferencial
; y[i+1] = y[i] + H * v[i]
; temp1 := H * v_prev
ld hl,h_euler
ld de,mul_16x16_32_factor1
ld b,0
ld c,2
ldir
ld hl,v_prev
ld de,mul_16x16_32_factor2
ld b,0
ld c,2
ldir
call mul_16x16_32
ld hl,mul_16x16_32_result+1
ld de,temp1
ld b,0
ld c,2
ldir
; y := y_prev + temp1
ld hl,(y_prev)
ld bc,(temp1)
add hl,bc
ld (y),hl
; mostramos los resultados:  "v=VALOR_V y=VALOR_Y"
ld de,v_prefix_string
ld c,9
call 5
ld hl,v
call show_fixed
ld de,y_prefix_string
ld c,9
call 5
ld hl,y
call show_fixed
ld de,nl_string
ld c,9
call 5
; hacemos v_prev = v
ld hl,v
ld de,v_prev
ld b,0
```

```
    ld c,2
    ldir
    ; hacemos y_prev = y
    ld hl,y
    ld de,y_prev
    ld b,0
    ld c,2
    ldir
    ; comprobamos si hemos alcanzado la última iteración
    pop bc
    dec bc
    push bc
    ld a,b
    or a
    jp nz,simulation_loop_start
    ld a,c
    or a
    jp nz,simulation_loop_start
simulation_loop_end:
    pop bc
    ret

nl_string:
    db 0Ah,0Dh,'$'
; variables y constantes relacionadas con la simulación
constante_gravedad:
    dw 09CDh     ; 9.8
constante_muelle_div_masa:
    dw 0100h     ; 1  (Km / M)
constante_amortiguador_div_masa:
    dw 0100h     ; 1  (Ka / M)
h_euler:
    dw 0003h     ; 0.01 (una centésima de segundo)
v:
    dw 0000h
v_prev:
    dw 0000h     ; velocidad inicial = 0
y:
    dw 0000h
y_prev:
    dw 0100h     ; sometemos al sistema a una alteración inicial: y_prev = 1.0
temp1:
    dw 0000h
temp2:
    dw 0000h
temp3:
    dw 0000h
y_prefix_string:
    db " y=$"
v_prefix_string:
    db "v=$"
```

Como se puede ver, la rutina es muy sencilla y sólo hay que organizar las operaciones teniendo en cuenta la precedencia de operadores. Si nos fijamos en la variable "y_prev", ésta está inicializada al valor 1.0 al arrancar la simulación: hacemos esto para forzar una alteración inicial en el sistema que provoque una autooscilación. Al final el sistema de estabiliza en y = 9.53.

CONCLUSIONES

La aritmética de punto fijo nos permite realizar cálculos con números reales en procesadores sin unidad de cálculo en punto flotante. El formato utilizado de Q8.8 deja bastante que desear desde el punto de vista de la precisión lo he usado para facilitar la comprensión de la materia. En una próxima entrega trataremos de utilizar estas mismas rutinas para implementar una transformada de Fourier o una transformada de Hartley en ensamblador.

Gracias a Armando Pérez por su ayuda con las rutinas de multiplicación. Todo el código, así como información adicional pueden obtenerse de la página oficial de la revista (*http://callmsx.gabiot.com*). Mis capacidades como programador de ASM son bastante mejorables: ruego me disculpen si no he implementado de forma más eficiente algunas rutinas. He procurado que en el código ASM primara la comprensión antes que la eficiencia. Cualquier sugerencia o corrección será bienvenida en *avelino@canarias.org*.



RC707
Konami's Mahjong
Konami 1984
32kb ROM

No podía faltar el clásico juego de tablero japonés dentro del catálogo de Konami. Una versión simple dentro de una ROM de 32 kb.

El juego

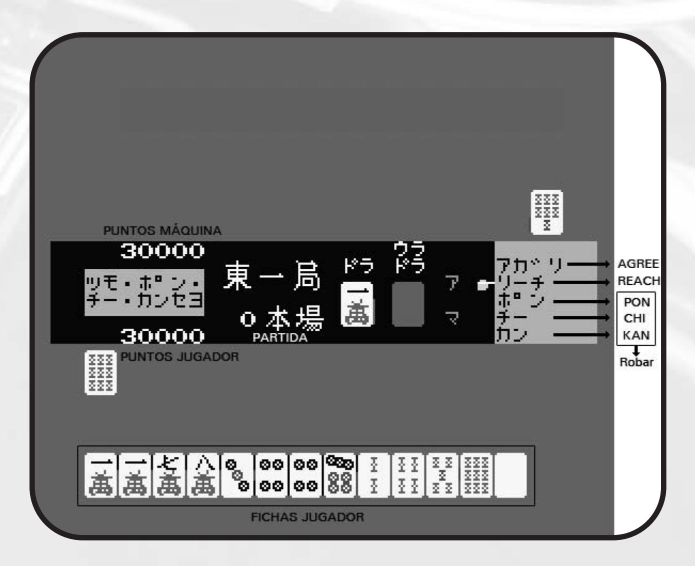
Al comenzar la partida podemos escoger entre tres niveles de dificultad que son el Amachua (Amateur), semiprofessional y professional. Tras la típica y corta pero animada melodía, aparece un tapete verde sobre el que van colocando las fichas de cada uno de los jugadores (13 fichas para cada uno, coincidiendo con los meses del calendario lunar) y en medio de éste se imprime diversa información sobre el estado de la partida además de un menú para escoger las acciones a realizar en una tirada.

Hay que destacar que el Mahjong originalmente fue pensado para cuatro jugadores (representando cada uno de ellos un punto cardinal) y que en este caso solo pueden jugar dos: el jugador contra la máquina; ni siquiera pueden jugar dos personas, una contra otra, algo que le quita muchos puntos a este cartucho. También se ha suprimido la elección del jugador que debe ser el viento del Este, aunque tratándose de dos jugadores tampoco tendría mucho sentido.

El objetivo del juego es hacer un Mahjong: esto se consigue cuando un jugador, despues de robar una ficha, tiene 14 fichas distribuidas en cualquiera de las combinaciones que se consideran Mahjong. Dicha combinación se reparte de la siguiente manera: cuatro combinaciones cualesquiera (pung o chow) y una pareja de fichas idénticas; aunque existen otras combinaciones excepcionales. En cada turno robamos una ficha de la muralla, teniendo temporalmente en mano 14 fichas y descartando siempre una si no se obtiene Mahjong.

El manejo y la información en tablero

Para jugar al juego se utilizan los cursores, la barra espaciadora y la tecla select; ésta última nos permite ejecutar las jugadas que podemos seleccionar en el menú, durante nuestra tirada. El tablero central nos muestra diversa información, como muestra la siguiente imagen.



Durante nuestro turno debemos coger una ficha y descartar una; también podemos realizar una de las jugadas, que son: Agree (aceptar), Reach (prepararse para completar la combinación de 14 fichas) y Pon / Chi / Kan (diferentes jugadas para robar fichas).

No comentaremos las reglas del Mahjong pero para los interesados en su funcionamiento pueden visitar (en inglés): *http://learnmahjong.brinkster.net/reachmahjong/intro.htm*

El juego cumple minimamente con su cometido. Gráficamente es bastante aceptable y el sonido se limita a unos pocos efectos.

Software Amateur

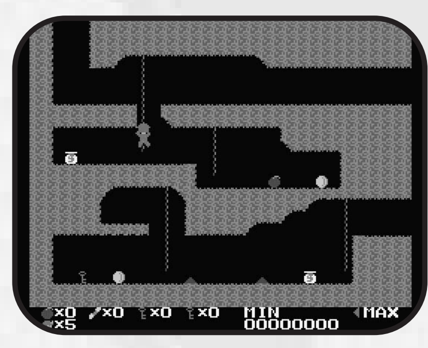
Mira por donde volvemos a encontrar material para difundir en esta sección. Esta vez hemos tenido donde elegir, ya que hemos encontrado bastantes programitas interesantes. Así que ya sabéis, si os llama la atención lo que leéis sólo tenéis que buscar la dirección que os apuntamos y a descargar se ha dicho. En esta ocasión todos los títulos vuelven a ser japoneses, excepto una demo holandesa.



SPELUNKERS

Casa : N.I., 1997
Música : PSG
Tipo: Aventuras
Sistema: MSX2

Spelunker es un clásico del videojuego. Creado en 1985 por la conocida empresa Irem, tuvo 2 partes en los salones recreativos. En 1986 salió la versión MSX1 programada por Broderbund pero que no buscaba una fidelidad absoluta con la original. Digamos que aunque tuviera el mismo concepto se trata de otra versión diferente con el mismo nombre. Además carecía de las virtudes que hacían de este juego especial. La velocidad del personaje y el extraordinario scroll se perdían y hacían de este título una triste versión.



El grupo N.I. nos presenta otra versión en Basic del mismo juego. Quizá nos recuerde más a la versión arcade en cuanto a la similitud del color pero tampoco busca esa perfecta recreación. En cambio de lo que sí dispone es de una velocidad envidiable y no peca del lamentable scroll de su antecesor. Aquí se pasa de una pantalla a otra mediante nueva creación.

El personaje se mueve muy bien, lo más difícil es el control en los saltos de las lianas, si no haces el salto preciso caerás y morirás en multitud de ocasiones, lo que hace que el juego no sea tan fácil. Por lo demás, disparamos, ponemos bombas para abrir nuevos caminos en la roca y cogemos llaves para ir abriendo puertas. Lástima que no tenga música de fondo, el único sonido que aparece es cuando nos matan una vida.
En fin, un juego que para pasar el rato no está mal, pero podía haber sido más



elaborado. Ya en 1986 habían juegos de la talla de The Goonies similares en concepto y con una diferencia notable con el de Broderbund. Esta versión última podía haber sido el Spelunker que el MSX merecía pero se ha quedado un poco corto. Así que si algún programador no sabe decidirse por hacer un remake, aquí tiene una buena oportunidad para dejar el listón más alto.

http://www.geocities.com/bixmatan/game1.html

MINI NIGHTS

Casa : N.I., 1997
Música : PSG
Tipo: Habilidad
Sistema: MSX2

Otro pequeño programa de la misma casa y que nos sorprende por su calidad.



Es un programa de una sencillez conceptual pero de una adicción brutal. Es un juego sin más pretensiones que conseguir que nuestro personaje que vuela por el espacio recoja todas las estrellas, aros y gemas en el menor tiempo posible. Esto ya lo habíamos visto dentro de juegos tan conocidos como el Super Mario o el Sonic, entre otros.

Lo primero de que nos damos cuenta es que tenemos una cuenta atrás en el tiempo y no podemos dormirnos en los laureles. Antes de que se agote debemos pasar por una plataforma para recuperar algo de tiempo para enfrentarnos a otra serie de estrellas. Es tipo Out Run, debemos superar tiempos parciales para poder continuar en la partida.



Tenemos tres “noches” para elegir, cada una con una disposición diferente de los elementos a escoger. Lo más difícil es hacer el mayor número de links, es decir, coger las máximas estrellas posibles en un único movimiento, lo que es bastante difícil pues el personaje se mueve muy rápido y es muy complicado de controlar cuando coge velocidad, ahí reside la gracia del juego.

Es una lástima que carezca de música, pues estaría muy bien que le acompañara algún tema de estos rápidos que se aceleran cuando el tiempo se va agotando. Tan sólo tiene un efecto sonoro.

Otro programita para pasar el rato. Este grupo japonés tiene muy buenas maneras pero no acaba de pulir sus programas, necesitarían un último esfuerzo para que sus juegos tuvieran un aspecto profesional.

CHACHA TO YAKKO NO VS MAGICAL COIN

Casa : Initialize, 1995
Música : FM
Tipo: Habilidad
Sistema: MSX2



Este es un juego que se ha puesto de moda entre los usuarios japoneses porque podemos encontrar unos cuantos en los últimos años. Puede ser debido a que su programación no requiere demasiada preparación o realmente porque es un juego adictivo y que gusta mucho por aquellas tierras.

¿Y de qué juego estamos hablando? En Europa se conoce como “Reverso”. En este juego participan dos jugadores quienes van colocando, cada uno en su



turno correspondiente, fichas en un tablero de 8x8 hasta cubrir todo el tablero o hasta que no se puedan colocar más fichas. El jugador que posee el turno podrá colocar una ficha en el tablero cuando al menos una de las fichas del oponente se encuentre en una línea horizontal, vertical o diagonal, entre una de sus fichas existentes y la posición donde quiere mover. Las fichas tienen la particularidad de poseer un color de cada lado. Cuando un jugador agrega una ficha al tablero, “todas” las fichas del oponente que se encuentran entre una ficha anterior y la recientemente ubicada (en todas las posiciones posibles), cambian su color al de las fichas del jugador de turno.

Pues el juego finaliza cuando hemos rellenado todo el tablero y se contablizan el número total de fichas que tiene cada jugador. El ganador será el que mayor número de fichas haya logrado colocar de su color.

Como vemos el juego es relativamente sencillo y para disfrutar más de él lo haremos con algún compañero como segundo jugador, ya que contra la máquina nos aburriremos pronto.

Los gráficos son coloristas, destacando la pantalla de presentación que cuenta con un dibujo manga muy llamativo. El tablero cumple pero podría ser más definido. Los personajillos de la derecha de la pantalla están bien, tienen animaciones según si lo hacemos bien o mal.

La única melodía del juego es lo mejor todo, está hecha para Msx Music y nos recuerda enormemente a la banda sonora del Twin Bee de Konami para Super Nintendo. La verdad es que está muy lograda, muy profesional.

Resumiendo, un programilla ideal para echarse un pique con un colega cuando no tengamos otra cosa mejor que hacer.

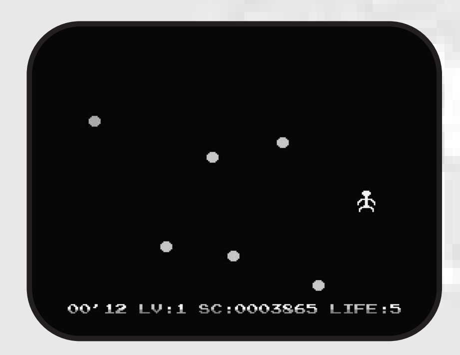
http://www.yashok.com/~yashok/msx/ctb/ctb_index.shtml

DANCING STARS

Casa : ZAP
Música : PSG
Tipo: Habilidad
Sistema: MSX2



Pequeño programa hecho en Basic que consiste fundamentalmente en esquivar todo lo que se te viene encima. Es un programa que está pensado para coger reflejos y mejorar nuestra habilidad en el manejo del teclado o del joystick.



En la pantalla de presentación podemos seleccionar la dificultad y diferentes tipos de handicap relacionados con los puntos matemáticos del juego.

A la hora de jugar llevamos un monigote (sprite minimalista donde los haya) y ya puedes empezar a "bailar". Que no te toquen las estrellas o las bolas según hayas seleccionado previamente. Cada nivel que superamos (cada veinte segundos) se va haciendo más difícil. Las estrellas cogen más velocidad y crece el número de ellas por la pantalla. Así hasta que logren tumbarte y el juego habrá terminado.

Y creo que no se puede comentar nada más, ni de gráficos ni de ¿sonido?. Un programita curioso, no dejes de probarlo.

http://www.kit.hi-ho.ne.jp/ zapzap/works_eng.htm

MERASH

Casa : Meraman, 1996
Música : FM
Tipo: Shooter
Sistema: Turbo R/ MSXDOS2

Merash es un shooter vertical, de esos que tenemos muchos en nuestro sistema. ¿Qué hace distinto Merash a otros matamarcianos? Lo primero es que no es un juego grande, no es el típico Zanac o Aleste en que vamos superando fases con el típico jefe de turno. El objetivo de este programa es hacerse el mayor número de puntos posible disparando a todo lo que se te viene encima, que no es poco.



Es muy difícil, disponemos "tan sólo" de cinco barras de energía. Es decir, a los cinco impactos morimos. Puede parecer mucho pero viendo la dificultad se hacen escasos.

Nos recuerda al Zanac en la forma de coneguir armas. Si disparamos a un monigote podremos conseguir cápsulas de poder, (L= láser, O= option, N=normal y P= puntos adicionales). Hacerte con el láser es lo mejor que te puede pasar si de verdad quieres aparecer en la tabla de récords, pues es el arma más eficaz.



Los gráficos son los típicos de los juegos del espacio. El diseño de las naves espaciales da el pego. El monigote le da un toque de humor al programa. El fondo es pobre, sólo es el negro espacio con algunas diminutísimas estrellitas que bajan en diferentes velocidades por el scroll. Se echa de menos gráficos de fondo.

La velocidad del juego es muy buena. Dada la cantidad de enemigos que aparecen en pantalla no da sensación de ralentización en ningún momento. Aquí queda demostrada la potencia del R800.

La melodía en MSX-Music es adecuada para el juego y tiene calidad, al igual que los efectos de sonido.

Que te apetece masacrar naves espaciales sin pretensiones, sin querer pasarte un juego largo, en Merash podrás hacerlo.

http://meraman.dip.jp/wiki.cgi?page= MsxDownload

MARBLE REVERSER

Casa : Studio Pikurusu, 1992
Música : PSG
Tipo: Puzzle
Sistema: MSX2, ratón

Como decíamos anteriormente estamos ante otro de los juegos de tablero llamado "reverso". En este caso nos sorprende la disposición del tablero que ya no es cuadrado sino en forma de hexágono y con las casillas también hexagonales a modo de panel de abejas. Nos será familiar si hemos jugado a juegos de estrategia tipo Risk. Además suman un total de 61 casillas, tres menos de lo que sería el tablero convencional.

La gran diferencia respecto al "Chacha to yakko no Vs Magical Coin"



reside en que Marble Reverser está programado en Screen 7 y tiene una definición gráfica mucho mayor. Por lo demás es el mismo juego pero aprovechando más el efecto diagonal. Se controla mediante ratón lo cual es más cómodo para acceder a picar en la casilla deseada.



La lástima es que carece de música y los efectos no son nada del otro mundo.

http://homepage1.nifty.com/youman/pxlsdown.htm

WAVES 1.2

Casa : Daniel Vik, 2006
Música : PSG
Tipo: Demo
Sistema: MSX1

Del coautor de la demo comentada en el anterior número, MSX Unleashed, y creador del fantástico emulador de MSX para Windows BlueMSX, nos llega otra demo para la primera generación de nuestro sistema: Waves, de Daniel Vik. Por su formato y tamaño, 512kb, sólo podemos ejecutarlo con Flashrom o con emulador. Aunque en anteriores versiones de su propio emulador se hacía necesario el uso de la emulación de MSX1, ahora es funcional también bajo modelos MSX2, aunque no en MSX2+ o turboR. Desde aquí recomendamos el uso de un MSX real, para disfrutar de la misma. En este último caso y como hemos adelantado, hará falta el uso de un cartucho flash o similar. Por suerte incorpora el mapper ASCII8, con lo que tanto LPE-FLASHG como la MegaFlashROM serán compatibles. Para su ejecución, DVIK utiliza la ESE RAM de ESE Artists' Factory.



La versión 1.2 de Waves es una actualización que desde la 1.0 inicial, incorpora diversas mejoras: arreglos de bugs en la colisión de sprites, mayor rapidez y suavidad en el scroll de texto que ahora pasa a actualizarse frame a frame, fallos gráficos, mayor compatibilidad en su ejecución en distintas máquinas MSX...

Para comenzar, el logo de DVIK en SC2 da paso a una interesante demo que deja descubrir dos agradables efectos: Por un lado, gráficamente descubrimos un split screen que divide una parte en Screen 2 (donde se aloja el título "WAVES" y el texto) de otra en screen 3, donde encontramos los gráficos. Por otra parte y referente al sonido, nos encontramos con música sampleada en PSG, gracias a un driver desarrollado por el propio DVIK y por ARTRAG. Sí, ¿quién iba a decir que 23 años después de su lanzamiento íbamos a disfrutar de un sonido tan limpio al reproducir música sampleada en PSG?





La parte superior, en Screen3, irá cambiando los gráficos (círculos concéntricos, una onda senoidal que disminuye en amplitud en su parte izquierda, espirales, personajes de la saga South Park, etc.) mientras realiza diversos efectos dignos de mención sobre ellos: superposición de elementos, rotaciones, cambios de color, etc.

La parte inferior, en Screen2, muestra durante toda la demo el clásico "scrolltext" donde el autor explica algunas características de la demo y agradece a diestro y siniestro a sus colaboradores. Además de esto, añade como ya se ha comentado el nombre de la demo "WAVES". Éste permanecerá inmóvil la mayor parte de la presentación, salvo unos segundos en los que sufrirá un efecto de onda, similar al ya visto en Dahku cuando se iniciaba la partida.

La parte sonora es digna de mención por segunda vez. Música sampleada en 8 bits para el uso simultáneo de los 3 canales del PSG. La frecuencia soportada es de unos 8KHz y al parecer la melodía está convertida de un WAV originario del PC. Es un sample limpio; curioso, sin duda.

En definitiva, WAVES es una demo bien trabajada y que muestra efectos anteriormente no vistos en MSX1. Si tenéis 5 minutos y el Hardware necesario, no perdáis la oportunidad de disfrutarla.

http://www.bluemsx.com/demos/waves1.2.zip

Trucos y Pokes

METAL GEAR 2

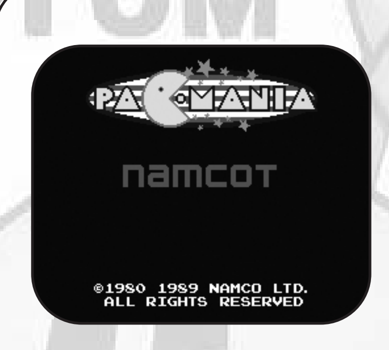


Test musical

Para poder escuchar las melodías del juego primeramente hay que empezar la partida y una vez en el juego, seleccionar los cigarrillos. Después, encendemos la radio con la tecla **"F-4"** y sintonizamos, con los cursores de izquierda y derecha, la frecuencia **140.07**, donde escucharemos las melodías del juego.



Para cambiar de melodía basta con cambiar la frecuencia a cualquier otra y volver a sintonizar la frecuencia **140.07**.



PACMANIA

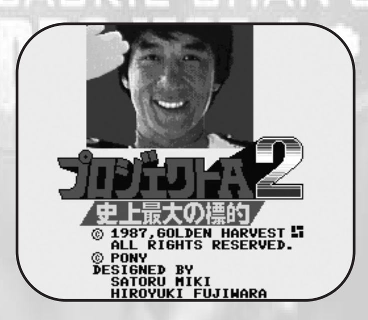
Seleccionar velocidad

Podemos variar la velocidad del juego si durante la partida pulsamos la tecla **"TAB"**.



Aparecerá un menú que nos permite escoger entre diez valores de velocidad (de 0 a 9). Seleccionamos el valor deseado con la teclas direccionales de izquierda / derecha y pulsamos la barra espaciadora para confirmar.

PROJECT A2



Vidas extras

Podemos aumentar nuestro número de vidas hasta 99 si pulsamos simultáneamente las teclas **"D"** y **"F"** durante el juego. Cada vez que pulsemos esta combinación, obtendremos una vida extra.



Recuperar la energía

Si estamos a punto de morir, podemos recuperar toda nuestra energía si pulsamos simultáneamente las teclas **"F"** y **"E"**, durante el juego. Podemos hacer este truco tantas veces como queramos.

NINJA ASURA



99 Vidas

Si queremos ver 99 vidas en nuestro marcador hay que pulsar la tecla **"ESC"** para parar la acción, después hay que pulsar simultáneamente las teclas **"SHIFT"** y **"F-2"**, así aumentaremos el número de vidas disponibles y si mantenemos esta combinación pulsada, rápidamente obtendremos las 99 vidas.

Para volver al juego hay que pulsar la tecla **"RETURN"**.





PENGUIN ADVENTURE

Continuar

Si queremos disponer de la opción de continuar la partida tras acabar ésta, hay que teclear seguidamente **"N"**, **"O"**, **"R"**, **"I"**, **"K"** y **"O"**, durante el menú de selección del nivel.

Cuando acabe nuestra partida bastará con pulsar la tecla **"F-5"** para continuar.

F5 KEY CONTINUE
GAME OVER

HISCORE 003390
SCORE 003390

Si hacemos lo mismo pero pulsando las teclas **"K"**, **"A"**, **"Z"**, **"U"**, **"M"** y **"I"**, podremos continuar la partida, pero con la diferencia de que perderemos todos los objetos.



Final feliz

Para poder reunirnos felizmente con la princesa pingüina hay que pausar el juego un determinado número de veces, siguiendo esta regla:

(4 x n) + 1

(4 x 0) + 1 = 1 vez
(4 x 1) + 1 = 5 veces
(4 x 2) + 1 = 9 veces

Es decir, para ver el final feliz hay que pausar el juego 1, 5, 9 o 13 veces. Cada vez que continuamos la partida el contador se pone de nuevo a 0.

Atajos

Para acceder a los atajos tenemos que dejarnos caer en una de las grutas en las que se esconda un atajo y pulsar la tecla direccional de abajo.

Pantalla 1 - Distancia 237, salto a pantalla 6



Pantalla 6 - Distancia 145, salto a pantalla 9



Pantalla 9 - Distancia 335, salto a pantalla 12



Pantalla 13 - Distancia 355, salto a pantalla 15



Pantalla 15 - Distancia 78, salto a pantalla 18



Pantalla 18 - Distancia 418, salto a pantalla 21



A partir de la pantalla 21 no encontraremos ningún atajo más. La imagen de la derecha muestra como queda el mapa después de coger todos los atajos.



Trucos y Pokes

PARODIUS
Códigos secretos



Para introducir los códigos secretos hay que pausar el juego pulsando la tecla **"F-1"**, introducir el código, pulsar la tecla **"RETURN"** y finalmente volver al juego pulsando de nuevo la tecla **"F-1"**.

PARO
Todas las campanas serán de color blancas.
KONAMI
Las campanas serán de color rojo.
BUTAKO
Las campanas siempre serán verdes.
TAKO18TH
Todas las campanas serán azules.
KATAI
Nuestro escudo protector resistirá el doble de impactos.
ZENBU
Obtendremos el máximo equipamiento. Solamente se puede hacer una vez por partida.



Fases secretas

Podemos encontrar tres fases secretas a lo largo del juego.

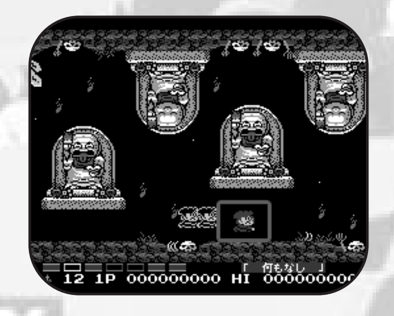
Nivel 1: el acceso secreto se encuentra detrás de la estatua del Moai.



Nivel 4: podemos encontrar la fase secreta al final de uno de los caminos que nos llevan a la parte superior de la pantalla, y que no tiene salida alguna.



Nivel 5: debajo de la décima tumba , situada en la parte de abajo, encontraremos la fase secreta.





PENGUINKUN WARS 2

Último nivel

Para luchar con el pingüino emperador sin necesidad de completar todos los niveles anteriores hay que caminar hasta llegar a la península helada situada al norte. Una vez allí, mientras estamos situados en frente del estadio de color azul, hay que mantener pulsada la tecla **"M"** y empezar a jugar con espacio.

I AM EMPEROR OF PENGUIN

Selección del contrincante

Dentro de cada fase encontramos tres contrincantes. Existe un truco que nos permite competir directamente con el contrincante que queramos. Para realizar el truco hay que dirigirse a la fase en la que quermos jugar; una vez situados al lado del estadio de dicha fase, hay que pulsar simultáneamente los siguientes números: **"2"** para el contrincante número 2, y **"1"** y **"2"** simultáneamente para el contrincante número 3; sin soltar estas teclas hay que empezar la partida pulsando una vez la barra espaciadora. Por ejemplo, si queremos jugar contra el último contrincante de una fase, tenemos que mantener pulsadas simultáneamente las teclas **"1"** y **"2"**, y pulsar una vez la barra espaciadora.

POLICE STORY, THE



Todos los objetos

Pulsar simultáneamente las teclas "**CTRL**" y "**F-1**", durante el juego. La primera vez que lo hacemos obtenemos el paraguas y el traje antibalas, la segunda vez obtenemos los guantes y los zapatos, con la tercera vez conseguimos la llave, con la cuarta el kit médico y finalmente, la quinta vez, el anillo.

Pasar de nivel

Para pasar de nivel hay que pausar el juego pulsando la tecla "**STOP**" y después hay que pulsar la tecla "**F-5**".



Aumentar la vitalidad

Para aumentar la vitalidad hay que pulsar simultáneamente las teclas "**CTRL**" y "**F-2**", durante el juego. Podemos subir nuestra vitalidad tantas veces como queramos.

Completar habitación

Para completar la habitación hay que pulsar simultáneamente las teclas "**CTRL**" y "**F-3**", durante el juego.

SA ZI RI



Test musical

Para acceder al test musical hay que pulsar la tecla **"F-5"** en el título del juego o durante la presentación. Hay 9 melodías que podemos ir escuchando si pulsamos la tecla **"X"**.



Parachuteless Joe

Juega una partida y haz una puntuación que te permita poner record. Escribe como nombre "*Palurdo*", para continuar jugando en el nivel que no consigas pasar o "*Puto Grajo*", para desactivar la aparicion del Grajo. Funcionan como ON/OFF, asi que volviendo a escribir lo mismo, vuelves a activar el Grajo, o desactivas los continues (ojo, abortar partida también desactiva el truco"*Palurdo*".) Verás que has activado algo si tras poner el código, abajo pone *CHEAT!*

STARSHIP RENDEZ-VOUS



Test musical

Para acceder al test musical que nos permite escuchar las 9 melodías del juego hay que pulsar la tecla **"F-2"** durante la introducción que podemos ver al insertar el primer disco.

Dentro del menú escucharemos las diferentes melodías utilizando la barra espaciadora.

Para salir basta con pulsar la tecla **"F-5"**.

Don't Warro! Be Japo!

Introduciendo el password **63AC F013**, empezarás a jugar en el nivel 51, hasta el 55. El juego normal terminaba en el 50.

EatBlue! 2004

Escribimos la clave **1973** para ir directos al nivel 11. Cuando se acaba el juego también se entrega el código para el Don't Warro! Be Japo! (63AC F013).

Trucos y Pokes

PSYCHO WORLD



Menú musical

Para acceder al menú musical hay que mantener pulsada la tecla **"M"** mientras carga el juego desde el disco.

Forzar sonido PSG

Podemos obligar la reproducción de las músicas a través del PSG si mantenemos pulsada la tecla **"P"** durante la carga del juego.

Lógicamente este truco es combinable con el del test musical.



Intercambio de teclas

El mantener pulsada la tecla **"X"** mientras carga el juego hace que las funciones del botón A o barra espaciadora y el botón B o SHIFT, se intercambien. Estas dos teclas se utilizan para saltar y disparar.



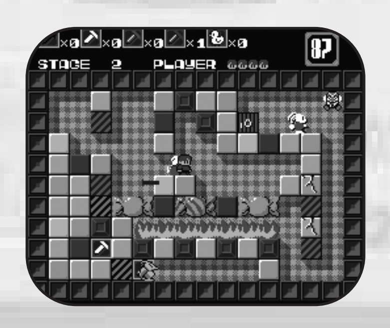
PROTECTOR, THE

Para saltar de nivel hay que pulsar simultáneamente las teclas **"T"** y **"O"**, durante el juego.

QUINPL



Para obtener inmunidad total hay que pulsar simultáneamente las teclas **"DEL"** (delete) y **"BS"** (backspace) cuando aparezca el logotipo de Bit2, y mantenerlas pulsadas hasta que aparezca el título del juego. Después comenzaremos la partida de manera normal, pero nuestro personaje nunca morirá.



RASTAN SAGA

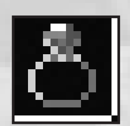
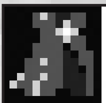
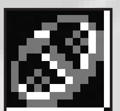


Obtener objetos

Para comenzar la partida ya equipados con uno de los objetos que podemos recoger durante el juego hay que esperar a que empiece la partida de demostración; durante la partida, esperamos a que el personaje obtenga uno de los objetos, en ese momento pulsamos la barra de espacio y comenzaremos la partida en posesión de ese objeto.



Los objetos que podemos obtener con este truco, son los que se encuentran en la parte superior derecha del marcador; podemos encontrar una capa, un anillo y un escudo.

RUNE MASTER II



Test musical

El menú musical esconde 31 melodías y efectos del juego. Para acceder a dicho menú hay que pulsar simultáneamente las teclas **"F-1"** y **"F-2"** en la pantalla de título del juego.

Con los cursores arriba y abajo seleccionaremos la melodía o efecto y con la barra espaciadora lo escucharemos.

Podemos salir del menú y volver al juego normal pulsando la tecla **"SHIFT"**.

Juego secreto

Podemos jugar a un juego de naves, que se encuentra oculto, si pulsamos seguidamente las teclas **"O"**, **"M"**, **"A"**, **"K"** y **"E"** en la pantalla del título.



RUNE MASTER III



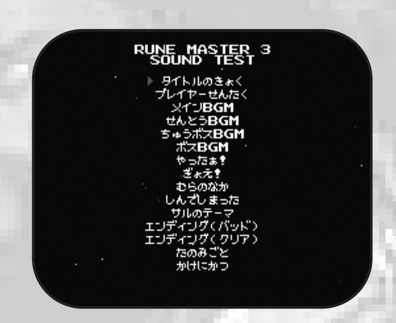
Test musical

Para acceder al menú musical que se encuentra oculto hay que arrancar el juego desde el disco número 2 mientras mantenemos pulsadas simultáneamente las teclas **"S"** y **"T"**.

El menú tiene 15 melodías (no todas las que suenan en el juego) que podemos seleccionar con los cursores direccionales de arriba y abajo.

Para reproducir la melodía hay que pulsar la barra espaciadora.

Podemos detener la reproducción de la música pulsando la tecla **"SHIFT"**.





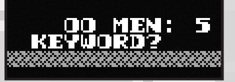
RISE OUT



Contraseñas

Para introducir las contraseñas debemos pulsar simultáneamente las teclas **"CTRL"** y **"P"**, durante el juego.

1 UNDR	**11** 5TOW
2 SNAK	**12** LAKE
3 BIGI	**13** GOLG
4 GRAS	**14** PYR2
5 LADR	**15** PYR3
6 2TOW	**16** OOBA
7 ASCI	**17** PYR1
8 LOOP	**18** PYR4
9 ROOM	**19** TRAP
10 AMID	**20** TANK



Vidas extra

Para obtener más vidas hay que comenzar la partida y durante el juego hay que pulsar simultáneamente las teclas **"CTRL"** , **"SHIFT"** y **"2"**. Cada vez que pulsemos esta combinacion obtendremos una vida extra. Podemos conseguir hasta 99 vidas.

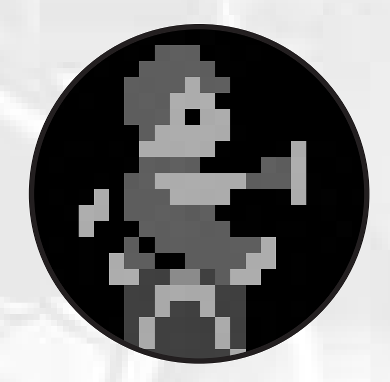
Pasar de nivel

Podemos pasar directamente al siguiente nivel realizando una combinación de teclas parecida a la del truco anterior.

Hay que pulsar simultáneamente las teclas **"CTRL"**, **"SHIFT"** y **"3"** para completar el nivel.

YOUKAI YASHIKI





EL JUEGO

Nos encontramos ante un gran juego de Casio para MSX de primera generación y superiores, a mi parecer el mejor que esta casa tiene. Un juego sin un gran nivel de dificultad, muy jugable y en el que con un poco de práctica se puede terminar en un par de horas (recomendado para todas aquellas personas con poco tiempo para jugar).

La historia comienza con un niño como personaje que debe atravesar un cementerio para adentrarse en una casa con una única linterna como arma con la que deberemos de alumbrar a nuestros enemigos para matarlos y poder rescatar a su novia. Una vez dentro, deberemos matar toda clase de fantasmas y monstruos, y recoger 5 objetos, los cuales están dispersos por toda la casa. Sin ellos no podremos llegar a ver el final del juego.

CÓMO JUGAR

El nivel de vida del jugador lo encontramos en la parte superior izquierda de la pantalla, en forma de pila. Al comenzar el juego dispondremos de 2 pilas, las cuales estan descargadas, por lo tanto el disparo de nuestra linterna no tiene apenas alcance, pero al conseguir los primeros 150 puntos, se recargarán las pilas y podremos disparar a larga distancia. Luego, conforme vayamos jugando, veremos que podemos conseguir más pilas y así tener más vida para poder matar a los monstruos.

El escenario (la casa), se divide en 5 zonas, cada una de ellas tiene su jefe final. Para llegar a él deberemos recoger 5 sellos con los cuales conseguiremos una llave. Una vez la tengamos deberemos enfrentarnos al monstruo de esa zona.

A lo largo de la casa, veremos unos pozos de los cuales sale fuego. Una vez se vaya el fuego, podemos saltar encima de ellos, y pulsando el cursor hacia abajo veremos que podemos aparecer en distintas partes de la casa, es decir, que hacen de funcion de *warp*.

Aparte, hay un sólo pozo por zona, el cual nos lleva a una habitación únicamente accesible desde éste. Ahí es donde nos encontraremos los objetos que debemos recoger para poder terminar el juego. De paso, también tendremos la oportunidad de poder recargar nuestras pilas, eso sí, sólo si somos hábiles con nuestra linterna.

Para no perdernos por la casa debemos tener en cuenta un par de cosas:

1.- El color de las plataformas: cada zona tiene un color distinto de plataformas.

2.- La música del juego: El juego tiene dos clases de músicas, una en la zona que debes matar al monstruo de turno y la otra música es la misma para las demás zonas.

MONSTRUO 1. LA BRUJA

Bastante fácil de matar, nos atacará con sus pelos de bruja (nunca mejor dicho), y nosotros deberemos ir disparándoles conforme nos vamos metiendo debajo de ella. Una vez estemos justo debajo de su boca, saltaremos y dispararemos en la boca. Con uno o dos disparos certeros la habremos matado.



MONSTRUO 2. LA CALAVERA

En mi opinión, para ser el segundo monstruo, es el más difícil de todos. Para poder llegar a él, una vez conseguida la llave, nos pondremos justo delante de la estatua que hay en la parte baja de esta zona y pulsamos el cursor hacia abajo para que se nos abra una trampi-



lla en el suelo y nos adentramos en ella.

Deberemos tener al menos 3 pilas para poder acabar con él (o que seas un as con los cursores). Nos disparará rayos, y nosotros para poder matarlo deberemos disparar a estos rayos, no a la calavera.

MOSTRUO 3. EL AVISPERO

Para poder acabar con él, deberemos subir a la parte superior de la escalera que tenemos delante nuestro. Le dispararemos justo en la parte superior que toca a la rama del árbol aunque tratará de evitarlo disparándonos nubes de avispas.



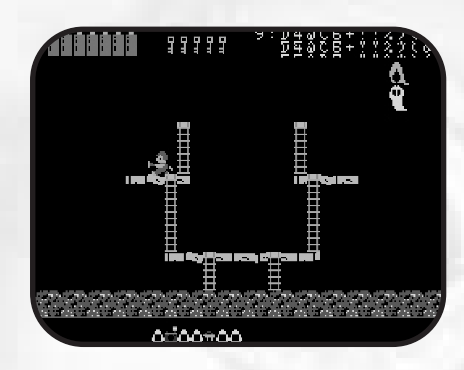
MONSTRUO 4. EL OJO QUE TODO LO VE

Bastante fácil también. Un gran ojo al que debemos disparar y muy fácil de evitar sus disparos.



MONSTRUO 5. LOS 4 FANTASMAS

Si os habéis dado cuenta, cada vez que matábamos a un monstruo, veíamos cómo se convertía en un fantasma que desaparecía del escenario. ¿Dónde iban a parar?, pues justo aquí. Tenemos a los cuatro apareciendo y desapareciendo por la pantalla. Para poder matarlos debemos ser ágiles con los cursores y dispararles justo cuando aparecen.



Tendremos que armarnos de paciencia ya que desaparecen muy rápido y es complicado acertarles. La ventaja que tenemos reside en que ellos no nos disparan, sólo se limitan a volvernos loco detrás de ellos. Con un disparo por fantasma abremos acabado con ellos.

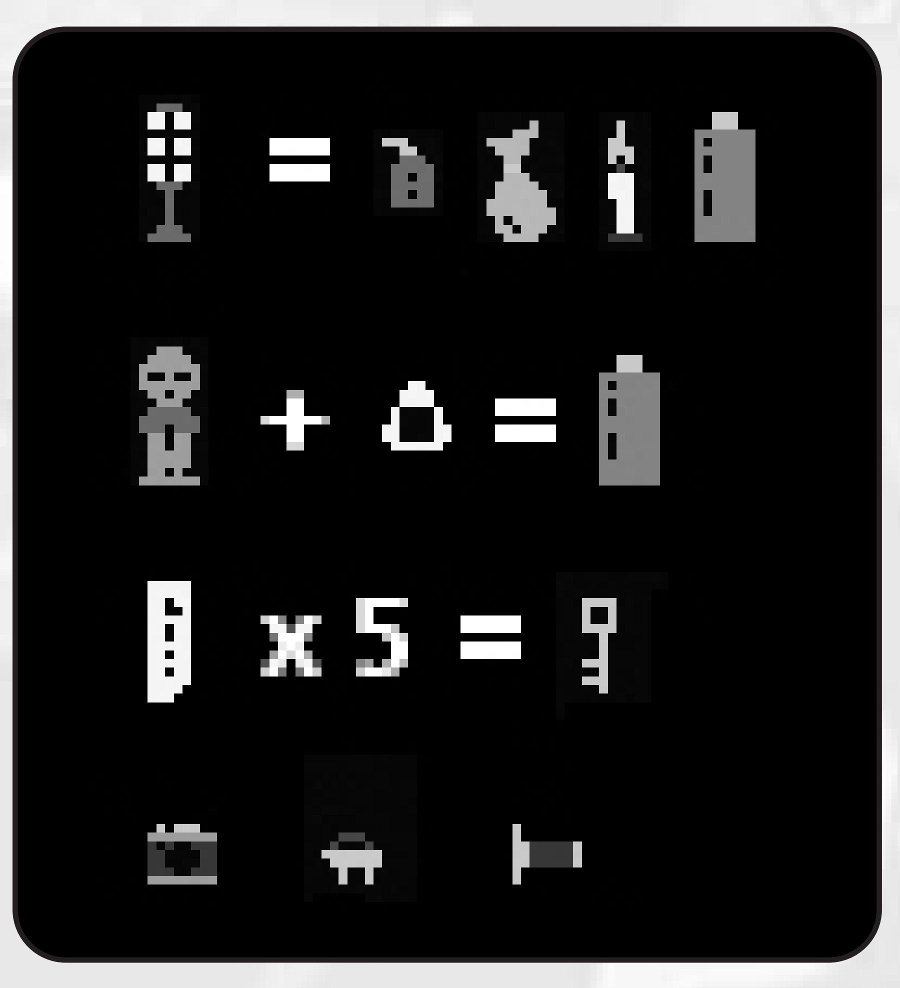
RECOMENDACIONES

Como podemos recorrer libremente todo el mapeado del juego, es muy fácil el ir pillando los objetos que nos dan las llaves, lo mejor es que sólo cojamos los objetos de cada zona ya que están justo los 5 por zona. Si no lo hacemos así luego nos volveremos locos recorriendo el escenario completo en busca de estos objetos.

Otro consejo es que procuréis coger los 5 objetos necesarios para llegar al final del juego en orden. Es decir, que vayamos probando los pozos que hay en cada zona, para acceder a la habitación donde está. Si no, luego no nos acordaremos en que zona cogimos el objeto y en cuál no.

Juan Miguel Ortuño Sánchez

OBJETOS



MegaFlashRom SCC

- Cartucho flash de 512k (4Mb) con SCC incluido.
- Soporte multirom: carga mediante menú de arranque.
- Compatibilidad Konami de 2 cartuchos simultáneos, incluso con Game Master 2.
- Detección y parcheo automático: compatibilidad con todo mapper conocido.

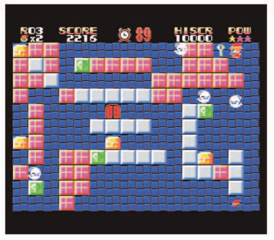
multi ROM collection

¡¡¡ 8 juegos de MSX1 en un solo cartucho!!!

Acceso secuencial de carga mediante RESET.

¡¡¡ Es muy bonito, muy barato y es sumergible a 34m (o más) !!!





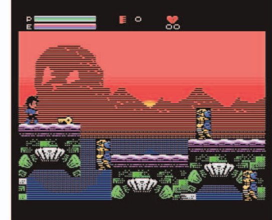


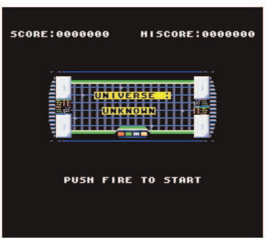








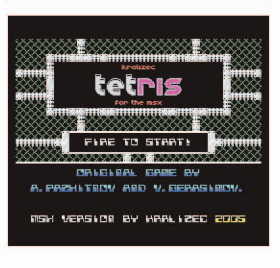








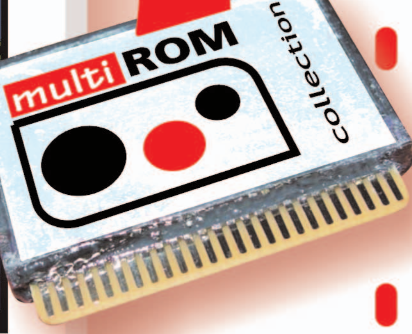












> http://soft.mundivia.es/mpazos/flashrom <